UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN: ESP. SISTEMAS DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

NORMALIZACIÓN Y MODELADO DE SERVICIOS WEB:

CATÁLOGO DE REFERENCIA, MODELO CANÓNICO DE DATOS Y ASISTENTE DE GENERACIÓN DE CONTRATOS DE SERVICIO

Autor: ***ALEJANDRO LABRADO BERMÚDEZ***

Tutor: ***ROMÁN LÓPEZ-CORTIJO GARCÍA***

Leganés, Octubre de 2015

TÍTULO: *Normalización y Modelado de Servicios Web: Catalogo de Referencia, Modelo Canónico de Datos y Asistente de Generación de Contratos de Servicio*

AUTOR: *Alejandro Labrado Bermúdez*

TUTOR: *Román López-Cortijo García*

La defensa del presente Proyecto Fin de Carrera se realizó el día **22** de **Octubre** de **2015**.

Siendo calificada por el siguiente tribunal:

PRESIDENTE	**Antonio de Amescua Seco**
SECRETARIO	**Luis García Sánchez**
VOCAL	**Germán Dugarte Peña**

Habiendo obtenido la calificación:

CALIFICACIÓN	

Presidente **Secretario** **Vocal**

“A mi familia, por navegar conmigo
esta última ola… “

Agradecimientos

Hace casi una década que comencé mi andadura profesional, ilusionado y con la emoción del que comienza a escribir un nuevo capítulo de ese enorme libro que es la vida. Las líneas fueron saliendo una tras otra y, aunque la mayoría eran alegres, siempre había un hueco vacío al volver la vista atrás.

Ahora ha llegado el momento de ir rellenando los huecos de ese libro, que el escritor no afrontó en aquel tiempo, pero que son imprescindibles para finalizar esa obra en la que no solo él ha puesto esfuerzo.

En primer lugar, quiero agradecer a Román, mi tutor, por haberme proporcionado la oportunidad de elaborar este proyecto y, sobre todo, por su honestidad. Sinceramente, espero volver a encontrarlo en las sendas que el futuro nos depara.

A mis padres, Alejandro y Eva, sin quienes no habría sido posible llegar hasta aquí, y a Evita, mi hermana mayor, por su comprensión y compañía. Por esto, y por la fe que tuvisteis en mí, muchas gracias de todo corazón.

A todos los compañeros de ese increíble viaje, los que me acompañaron en cada uno de los caminos y los que me guiaron por la dirección correcta, gracias por haber llenado mi vida de momentos inolvidables, siempre tendréis un sitio en mi pensamiento.

Finalmente, gracias a todas las personas que me han acompañado en estos últimos años, los primeros de mi vida laboral, quienes me han hecho crecer personal y profesionalmente; una mención especial a esos grandes profesionales que han confiado en mí, han apostado por mí en esta lotería ofreciéndome oportunidades que probablemente no merecía, espero no haberlos defraudado y que estemos ganando todos.

Y, en definitiva, a todos los que se alegran por mí en este momento. GRACIAS.

"Disfruta de las pequeñas cosas,
porque puede que un día vuelvas la vista atrás
y te des cuenta de que eran las cosas grandes"
Robert Brault.

Resumen

La construcción de sistemas de software a escala empresarial es una tarea compleja. Las grandes corporaciones apoyan sus desarrollos aplicativos en arquitecturas orientadas a servicios (*SOA – Service Oriented Architecture*): SOA proporciona un modelo conceptual que da respuesta a las necesidades de negocio mediante la implementación de soluciones basadas en funciones auto-contenidas, independientes del contexto y de la infraestructura y con interfaces claramente definidas. Estas funciones, implementadas en una o varias tecnologías, son denominadas servicios y son expuestas para ser utilizadas por el resto de áreas de la organización, por terceros o incluso para formar parte de otros servicios.

Este Proyecto Fin de Carrera propone la normalización de los servicios mediante la utilización de entidades canónicas en la definición de los contratos, de modo que, todas las áreas de una organización utilicen un mismo modelo de datos para definir sus mensajes.

Esta normalización se apoya en la utilización de un registro centralizado, que almacena el glosario de campos, el catálogo de entidades canónicas y los contratos de servicios web (*WSDL – Web Service Description Language*), así como todos los metadatos necesarios para facilitar la explotación y la gobernabilidad de los diferentes elementos.

Adicionalmente, se presenta una solución integrada con el entorno de desarrollo Eclipse (*IDE – Integrated Development Environment*) que facilita tanto el mantenimiento del glosario de datos centralizado, como la definición de entidades canónicas y modelado de contratos de servicios web.

Esta herramienta de ayuda al desarrollo facilita el diseño de las interfaces de los servicios web, a la vez que garantiza el cumplimiento de la normativa impuesta por la arquitectura, basada en unos patrones de diseño previamente definidos.

El resultado de la normalización de los servicios es una clara mejora en la eficiencia de los proyectos, suponiendo una reducción notable de los costes de integración; la utilización de herramientas de registro facilita el gobierno de los servicios y, con la extensión de la misma a través de herramientas de modelado, se evitan errores de definición muy comunes en arquitecturas orientadas a servicios.

Abstract

Building an enterprise-scale software system is a complex task. Large corporations base their applications developments in service-oriented architectures (*SOA - Service Oriented Architecture*): SOA provides a conceptual model that responds to business needs implementing solutions based on self-contained functions, context and infrastructure independent and with a clear defined interfaces. These functions, implemented in one or more technologies are called services and they are exposed to be used by other areas inside the organization, by third parties or even as a part of other services.

This project proposes the standardization of services using canonical entities to define contracts, in order that every area in an organization uses a common data model to define their messages.

This standardization is based on using a centralized register, which stores the data dictionary (fields glossary), the canonical entities catalogue and web services contracts (*WSDL - Web Service Description Language*) as well as all necessary metadata to facilitate the operation and governance of different elements.

Additionally, an integrated solution with the Eclipse development environment (*IDE – Integrated Development Environment*) is given to facilitate the centralized data glossary maintenance, canonical entities definition and web services contracts modelling.

This development aid tool facilitates the Web services interface design, while it ensures compliance with the rules imposed by the architecture, based on a predefined design patterns.

The result of the standardization of services is a clear improvement in terms of projects efficiency, assuming a significant reduction in integration costs; the use of registry tools facilitates the services government and the extension of itself through modelling tools where common definition errors on service-oriented architectures are avoided.

Índice

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

"La especie más fuerte no es la que sobrevive,
tampoco la más inteligente,
sino la que se adapta mejor al cambio"

Charles Darwin.

En un mundo tecnológico de constantes cambios, se espera que las áreas de Tecnología de la Información (TI) de las empresas respondan rápida y eficientemente a las necesidades del negocio.

La implantación de una Arquitectura Orientada a Servicios (en adelante, *SOA*) proporciona la flexibilidad que las organizaciones necesitan para el desarrollo y evolución de sus aplicativos, permitiendo la evolución de su modelo de negocio hacia la tercerización y facilitar la integración de sistemas/aplicaciones de diversos proveedores, a través de la generación de servicios: funcionalidades atómicas, altamente reutilizables y agnósticas de la plataforma que los soportan.

La gran problemática de las arquitecturas orientadas a servicios reside en la gobernanza de los mismos, cuyo objetivo es evitar inestabilidades que provoquen justo lo contrario de lo que se pretende; diversos expertos en el área SOA afirman: "*Pretender desarrollar una* ***Arquitectura Orientada a Servicios*** *(SOA) sin la implementación de un* ***Gobierno SOA,*** *es equivalente a intentar construir un edificio sin cimentación"*.

Este Proyecto Fin de Carrera pretende diseñar una serie de utilidades de arquitectura que ayuden a aplicar las políticas de gobierno de los servicios de una organización. En particular, aquellas que persiguen:

1. Asegurar que los servicios que se identifican y construyen pueden ser consumidos por el resto de la organización.
2. Evitar duplicidades de servicios o solapamientos de su funcionalidad.
3. Acordar un lenguaje común para los mensajes de los servicios.

En primer lugar, se trata de diseñar un modelo capaz de garantizar que los datos o parámetros de entrada/salida de los servicios estén bien definidos y sin duplicidades, que facilite y agilice el trabajo de los analistas, técnicos y de negocio, a la hora de definir los procesos a implementar, intrínsecamente definidos en los contratos de los servicios.

Por otra parte, el segundo objetivo de este proyecto consiste en proporcionar una herramienta de modelado de interfaces de servicios web en particular, parametrizable a través de ficheros de propiedades. Para la implementación de dicha utilidad es necesario definir unos patrones de diseño y ordenación de los contratos de servicio. El patronado de los servicios es especifico de cada organización, y más en particular de cada tipología de negocio (negocios del mismo sector tienen necesidades parecidas), por lo que en el desempeño de este proyecto se tomará como modelo una entidad bancaria con una arquitectura específica.

Para la consecución de los objetivos anteriormente mencionados, este Proyecto Fin de Carrera se estructura en 6 capítulos, cuyo contenido se resume a continuación:

- En el *Capítulo 1* se describe de manera general la labor realizada en el presente Proyecto Fin de Carrera, además de los objetivos que se persiguen con su realización. También se incluye una breve descripción de los capítulos de esta memoria.
- El *Capítulo 2* plantea los conceptos teóricos necesarios que permiten entender la situación objetivo y el camino seguido para su consecución.

  En primer lugar, se presenta una introducción a las arquitecturas orientadas a servicios dentro del mundo empresarial y la problemática que conlleva su aplicación.

  A continuación, se asientan las bases teóricas del modelo que implementa este proyecto enfocado a la normalización de los mensajes de los servicios: glosario de datos, modelo canónico, ordenación de servicios...

  Finalmente, se propondrá un modelo conceptual como solución para la normalización de los servicios (enfocándose en un modelo bancario), así como para el modelado de aquellos con naturaleza web de acuerdo a ciertos patrones.
- En el *Capítulo 3* se presenta el planteamiento técnico del *Catálogo de Referencia*, que soportará tanto el glosario de datos y entidades lógicas como el catálogo de servicios. Diferentes secciones detallan a alto nivel la solución y los elementos que intervienen en la misma: selección de tecnología, modelo conceptual del Catálogo de Referencia, personalización de la instalación y capa de presentación.
- En el *Capítulo 4* se detalla la herramienta *Eclipse* (*plug-in*) que permitirá tanto del mantenimiento del Catálogo de Referencia como el modelado de interfaces de servicios web desde el entorno local del desarrollador. Se expone a alto nivel la estructura del aplicativo, el funcionamiento de sus módulos principales, su parametrización y su interfaz de usuario. Adicionalmente, se incluye el manual de referencia reducido para que un desarrollador pueda ponerse a desarrollar la utilidad y distribuirla.
- El *Capítulo 5* resume las conclusiones obtenidas tras la realización del presente proyecto y plantea una serie de líneas futuras de trabajo a desarrollar,

estrechamente relacionadas con el modelo propuesto, que vienen a mejorarlo y/o complementarlo.

- En el *Capítulo 6* se explica resumidamente la metodología de trabajo que se ha aplicado, basada en el concepto de *metodologías ágiles*. En este apartado también se realiza la valoración económica de la ejecución del proyecto, teniendo en cuenta cada una de las fases del proyecto y los costes asociados

Finalmente, se incluyen una serie de documentos anexos que detallan el proceso de instalación de la solución y las guías de uso de la misma; también se incluye como anexo un documento introductorio a la filosofía de trabajo ágil Scrum.

Capítulo 2

Estado de la Cuestión

Existen una serie de conceptos que el lector debe conocer para facilitar el entendimiento de las motivaciones de este Proyecto Fin de Carrera. Se hará una breve introducción de cada uno de dichos conceptos teóricos para, finalmente, exponer la situación objetivo de la solución.

2.1. Introducción a la Arquitectura SOA

La arquitectura orientada a servicios no es un concepto actual. Fue desarrollada a finales de los 90's, pero es en la actualidad cuando está tomando un papel más importante en el mundo empresarial, sobre todo debido a la expansión del uso de servicios web; aunque hay que dejar claro que SOA no está vinculado a una tipología de servicios en particular.

SOA (Arquitectura orientada a servicios) es un marco de trabajo conceptual que establece una estructura de diseño para la integración de aplicaciones, que permite a las organizaciones unir los objetivos de negocio, en cuanto a flexibilidad de integración con sistemas legados y alineación directa a los procesos de negocio, con la TI.

Para entender el enfoque SOA, se debe tener claro qué es un servicio. Existen múltiples definiciones conceptuales y no existe consenso al respecto, pero puestos a elegir una definición concisa y completa:

- Según IBM, "*un **servicio SOA** es una especificación de una determinada lógica de negocio encapsulada, estándar, reutilizable, desacoplada e independiente de la tecnología, que se pone a disposición de la organización".*

NOTA: Los servicios web son la forma más conocida de implementar los servicios de SOA, pero **SOA no son *webservices*** y puede ser implementada sin ellos.

En el contexto actual de los mercados, en que la fusiones e integraciones de sistemas se producen cada día, la arquitectura SOA proporciona un enfoque capaz de aportar la flexibilidad que el negocio necesita. No se trata de cambiar los modelos organizativos por uno nuevo basado en SOA, sino de aprovechar los sistemas heredados, realizando servicios que se encargarán de actuar como fachada entre los antiguos sistemas y el resto de servicios.

Es por esto, que cada vez coge más fuerza dentro de las grandes corporaciones el concepto de **mediación de servicios a través de ESB's** (del inglés, Enterprise Service Bus), que facilitan la comunicación entre los servicios y ofrecen funcionalidades de valor añadido como alta disponibilidad, enrutamiento inteligente, orquestación de procesos, monitorización, etc.

Ilustración 1. Enterprise Service Bus

Como se ha podido deducir a lo largo de esta sección, en la arquitectura SOA existe un **factor fundamental** del que depende el éxito de una implantación: **el Gobierno SOA** es el encargado de velar porque se cumplan los principios de la arquitectura, así como de proporcionar las palancas necesarias para que así sea.

2.1.1. Principios SOA

Para comprender una arquitectura SOA, hay que conocer las características esenciales sobre las que se asienta. A continuación, explicaremos los principios más relevantes:

- **Contratos de servicio estandarizados**: la interfaz de entrada/salida (contrato con el cliente) tiene que estar explícitamente declarada, es decir, los campos que forman parte de este interfaz deben estar correctamente tipificados y ser conocidos. Estándares como WSDL y XSD hacen que el contrato del servicio este autodescrito.
- **Bajo acoplamiento entre los sistemas:** cuando menor es la dependencia entre servicios, mayor es la independencia para el diseño y evolución de los mismos.
- **Abstracción de la funcionalidad:** el servicio debe ser una caja negra, únicamente definido por su contrato, que a su vez es el mínimo acoplamiento posible con el consumidor del mismo.
- **Reusabilidad de servicios:** encapsulación de la lógica de negocio en servicios reusables por otros servicios.

- **Servicios sin estado:** todos los datos que necesita el servicio para trabajar provienen de los parámetros de entrada, en caso contrario se vería afectada la escalabilidad del mismo.
- **Capacidad de descubrimiento de servicios:** se refiere a la necesidad imprescindible de que nuestro catálogo de servicios esté disponible, publicado, accesible y adornado con una serie de meta datos que permitan lanzar búsquedas ricas sobre el catálogo para identificar la existencia de servicios que podamos reutilizar.
- **Composición de servicios basados en otros servicios (orquestación):** se refiere a la capacidad de implementación de nuevos servicios complejos a partir de otros ya existentes.
- **Autonomía:** el servicio tiene control sobre su entorno de ejecución y sobre la lógica que encapsula.
- **Transparencia de ubicación e interoperabilidad de los servicios:** se refiere a la capacidad de un consumidor para invocar a un servicio independientemente de su ubicación en la red y tecnología

2.1.2. Beneficios

A la hora de evaluar los principales beneficios de utilizar una arquitectura SOA en una organización, vamos a realizar una diferenciación entre los que son relevantes desde el punto de vista del negocio y los que aplican al ámbito tecnológico.

- Negocio:
  - ✓ **Eficiencia**. Transforma los procesos de negocio en servicios compartidos con un menor coste de mantenimiento.
  - ✓ **Capacidad de respuesta**. Rápida adaptación y despliegue de servicios, clave para responder a las demandas de clientes, partners y empleados.
  - ✓ **Adaptabilidad**. Facilita la adopción de cambios añadiendo flexibilidad y reduciendo el esfuerzo.
- Tecnológico:
  - ✓ **Reduce la complejidad** gracias a la compatibilidad basada en estándares frente a la integración punto a punto.
  - ✓ **Reutiliza los servicios** compartidos que han sido desplegados previamente.
  - ✓ **Integra aplicaciones** heredadas limitando así el coste de mantenimiento e integración.

2.1.3. Gobierno SOA

Es el organismo que establece y hace cumplir las políticas y procesos necesarios para controlar el ciclo de vida de los servicios, es decir, el organismo que vela por el cumplimiento de los principios SOA y, en última instancia, el responsable del mantenimiento y evolución de la arquitectura de la organización.

Para todo esto, el Gobierno SOA puede y debe valerse de las herramientas necesarias para gestionar y supervisar los servicios de la entidad, implantadas de acuerdo a sus propias políticas.

En el ámbito de este proyecto, se resuelve una parte crítica de la problemática de gobierno de los servicios mediante la implantación de una serie de utilidades:

- **Asegurarse de que los servicios que se identifican y construyen pueden ser consumidos por el resto de la organización.**
- **Evitar redundancia de servicios**: Para esto es muy necesario una herramienta que actúe de registro en tiempo de diseño para que los analistas puedan buscar, con criterios de negocio, si ya existe la funcionalidad que necesitan.
- **Acordar un lenguaje común de comunicación**: los parámetros de entrada/salida de los servicios deben estar bien definidos, ser únicos y estar alineados tanto con el modelo de datos de la organización como con las entidades lógicas de negocio.

Por todo esto, se puede afirmar que el concepto más importante en una estrategia SOA es el **Catálogo de Servicios**. Constituye la parte más tangible de la implantación de una estrategia SOA, ya que publica y recopila de forma ordenada los servicios de negocio y de infraestructura (servicios SOA) basados en estándares y ontologías, haciéndolos así disponibles para todos los sistemas de información de la organización, que pueden reutilizarlos tal cual cuando su funcionamiento requiera conocer un determinado evento de negocio o notificar un determinado evento de interés para el negocio.

2.2. Normalización de Servicios: Modelo Canónico de Datos

Por definición, normalización es: "*la actividad que tiene por objeto establecer, ante problemas reales o potenciales, disposiciones destinadas a usos comunes y repetidos, con el fin de obtener un nivel de ordenamiento óptimo en un contexto dado*".

La normalización de servicios consiste en la implantación de un idioma común en todas las áreas de una organización y que, además, esté alineado con el modelo de datos de la misma.

Si nos quedamos con esta visión únicamente, la existencia de un **glosario de términos común**, coherente con el modelo de datos de una organización, bastaría para normalizar los mensajes de comunicación de los servicios.

Sin embargo, la propuesta de un **modelo canónico de datos** consiste en llegar un paso más allá: es un modelo que define la estructura de la información en una organización, siendo su objetivo no solo el limitarse a modelar los datos de la misma, sino a servir de referencia para todas las entidades lógicas y sus relaciones a través de todas las bases de datos de la empresa y las aplicaciones legadas que tributen a la iniciativa.

Es decir, mientras que el glosario de términos identifica y define los atributos independientes que describen la información, el modelo canónico define su estructura e interrelaciones a nivel empresarial.

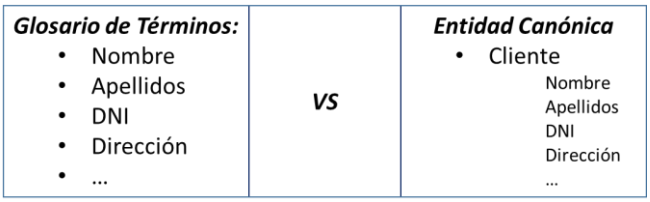


Ilustración 2. Glosario de Términos VS Modelo Canónico

Cada una de las estructuras lógicas representadas por un modelo canónico se denomina entidad canónica, es la unidad básica de definición del modelo, y son definidas con diferentes enfoques en función del contexto (*top-down*, *bottom-up*, estándares…).

El modelo canónico de datos puede ser utilizado para modelar los mensajes de los servicios, así cada aplicación sabrá qué información enviar/esperar en los mensajes. Este concepto está estrechamente relacionado con SOA y la estructura de los mensajes de los contratos de servicio.

Por tanto, podemos concluir que existen dos elementos clave para el correcto desarrollo de una iniciativa SOA: el primero, la creación de un **glosario de términos** y, el segundo, el desarrollo de un **modelo canónico de datos** que permita tener una visión global de los datos de la empresa. Ambos elementos constituyen, en conjunto, el **Diccionario de Datos** de la organización.

2.3. Topología de Servicios: Modelo BIAN

Todas las grandes corporaciones se encuentran con un problema común y bien conocido: la **excesiva complejidad de sus carteras de aplicaciones y servicios**.

Toda esta problemática se traduce en sistemas poco flexibles, no adaptables fácilmente, con costes de mejora, mantenimiento y operacionales elevados; en resumen, incapacidad para hacer las palancas necesarias para evolucionar rápidamente hacia soluciones avanzadas, nuevas tecnologías, métodos y modelos de negocio.

Los sistemas en muy pocas ocasiones se montan desde cero, sino que son producto de una evolución paulatina de modelos anteriores. En consecuencia, la problemática de la identificación y clasificación de los servicios no es un tema trivial; no se trata sólo de redundancia y solape funcional de los aplicativos, sino de problemas asociados a las integraciones punto a punto existentes entre ellos y la complejidad relativas a sus dependencias.

La identificación y clasificación de los servicios, y sus relaciones, es específica para cada negocio y las necesidades del mismo; sin embargo, es posible la **identificación de sinergias** entre organizaciones del mismo sector.

Para el sistema financiero existe una organización que propone un enfoque distinto para el problema descrito:

- **BIAN - Banking Industry Arquitecture Network**: Es una asociación de bancos, proveedores e instituciones educativas constituida con el propósito de definir un estándar de servicios para la industria financiera.

BIAN extiende el concepto general de diseño SOA mediante la identificación de particiones en base a capacidades funcionales genéricas de negocio, que puedan ser implementadas mediante servicios y representen los bloques de construcción de cualquier banco, para establecer un estándar.

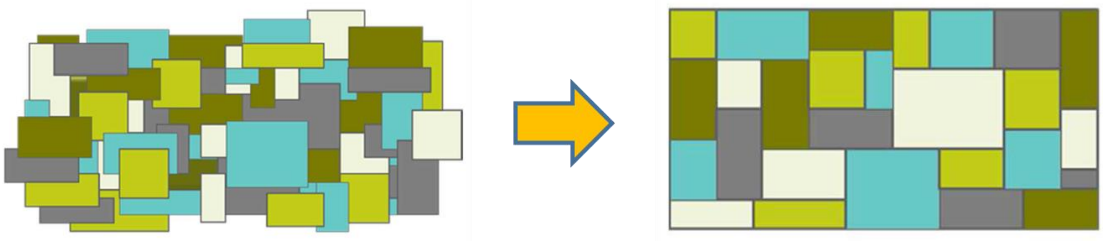


Ilustración 3. Planteamiento BIAN

Los diseños de BIAN son “canónicos”, es decir, pueden ser consistentemente interpretados por cualquier entidad en la mayoría de situaciones presentes en la misma:

para definir este tipo de diseño el enfoque debe ser diferente a todas las técnicas tradicionales basadas en procesos y lanzarse a una implementación más detallada.

BIAN define y mantiene un **mapa de referencia de servicios bancarios** agrupados jerárquicamente según los siguientes citerios:

- **Business Area** – es el nivel más alto de clasificación. Agrupa un conjunto amplio de capacidades de negocio. Para el *BIAN Service Landscape* son aspectos de la actividad de negocio que tienen necesidades de información específicas. Son 5: "*Reference Data*", "*Sales & services*", "*Operations & Execution*", "*Risk & Compliance*", "*Business Support*".
- **Business Domain** – en el siguiente nivel, los dominios de negocio definen una colección coherente de capacidades dentro de un área de negocio más amplio. Para el *BIAN service landscape* están asociados con skills y conocimientos reconocibles en el negocio bancario. En total existen 35 business domains.

  Ejemplos: "*Party*", "*Marketing*", "*Sales*", "*Customer Management*"…
- **Service Domain** – es el nivel más granular particionalmente, cada dominio define un conjunto único y discreto de capacidades de negocio (podría considerarse como una agrupación de operaciones). Los *services domains* son los bloques elementales de construcción de servicios. En total hay 280 service domains.

  Ejemplos: "*Party Data Management*", "*Party Profile*", "*Product Pricing Facility*"…

A continuación, se presenta un ejemplo de organización de servicio según BIAN:

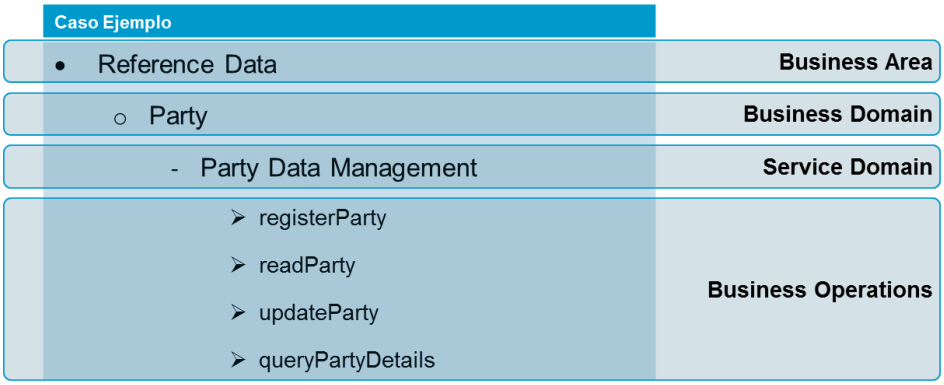


Ilustración 4. Ejemplo de clasificación de un servicio particular

Finalmente, el mapa de referencia de BIAN v4.0 considera que los servicios bancarios se clasifican en base a:

- 5 Business Areas (2 subareas)
- 36 Business Domains
- 280 Service Domains

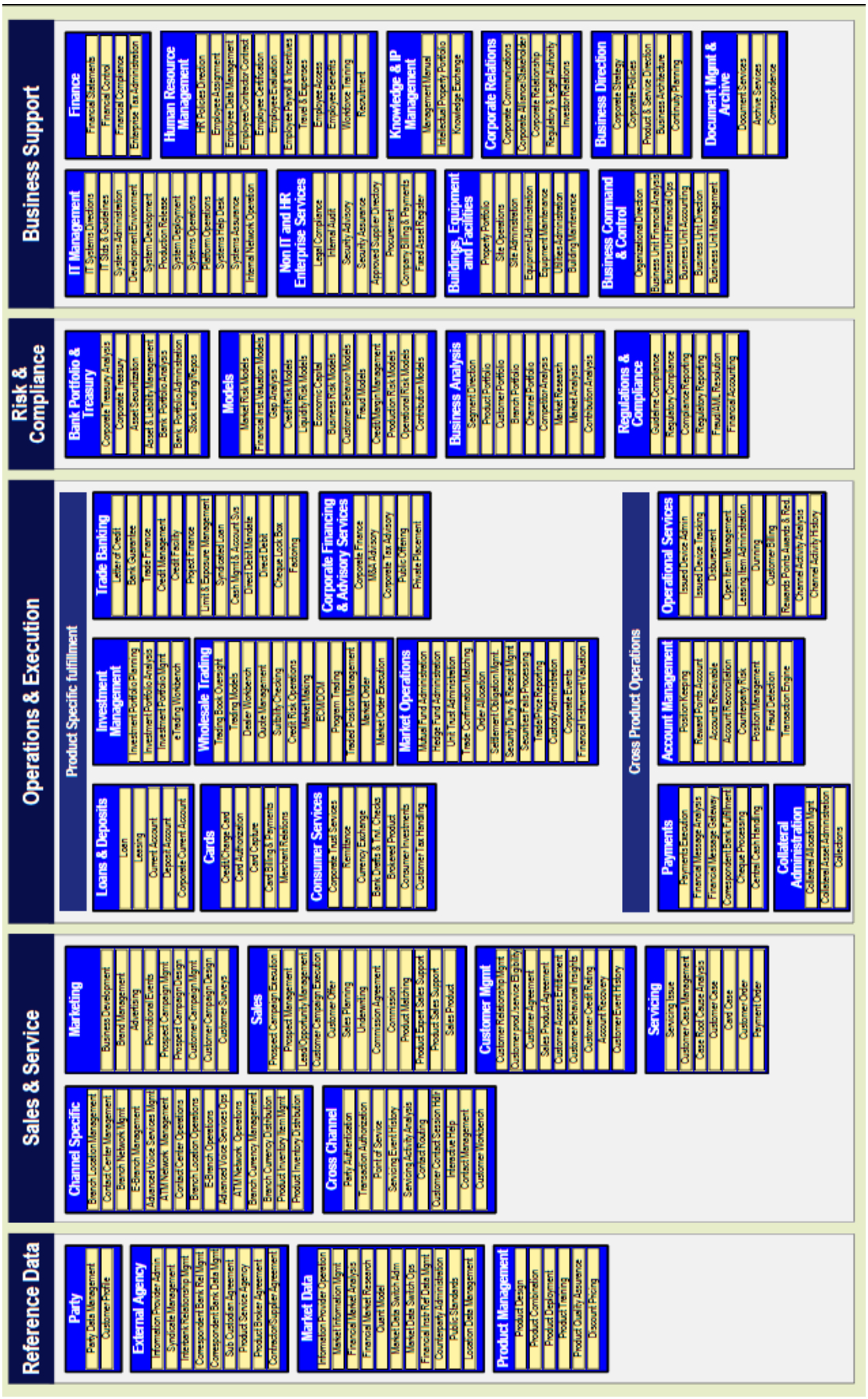


Ilustración 5. Landspace BIAN 4.0

Los principales beneficios que se logran con una organización de servicios basados en este modelo, son:

- Desarrollo e integraciones de software más eficientes.
- Oportunidad de crear soluciones más grandes y reutilizables.
- Adopción de modelos más flexibles de contratación de servicios de negocio y orientados a la externalización.

Además, BIAN no sólo no es incompatible con los estándares de mercado, sino que viene a complementarlos:

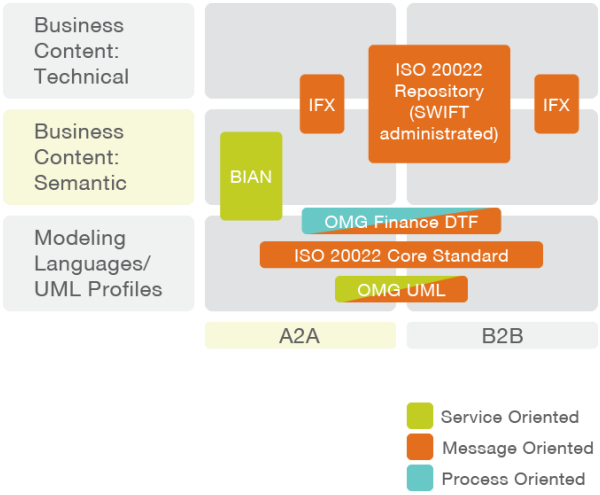


Ilustración 6. BIAN y otros estándares

NOTA: Esto es lo que hace que sea un modelo utilizable dentro de la solución que proponemos en el ámbito de este proyecto.

2.4. Modelado de Servicios Web

XML no es suficiente por si sólo para describir un servicio web, ya que es necesario asentarse sobre unos patrones con el objetivo de que los desarrolladores no tengan que investigar en las definiciones del esquema intentando descifrar cómo interaccionar con el servicio.

El lenguaje de descripción de servicios web (**WSDL – Web Service Description Language**) es un dialecto basado en XML que describe la forma de comunicación del servicio web: las operaciones, los mensajes, los requisitos del protocolo y los formatos de los mensajes. Además, dicho dialecto es extensible y se puede utilizar para describir casi cualquier servicio de red, incluyendo protocolos SOAP sobre HTTP, SOAP sobre JMS e incluso protocolos que no se basan en XML.

Por tanto, un documento *WSDL* es capaz de representar el contrato entre un servicio web y sus consumidores, es decir, proporciona la información necesaria al cliente para interaccionar con el servicio web.

Planteamiento conceptual de WSDL:

1.- La definición abstracta de puertos y mensajes se separa del protocolo de comunicación y del formato de los datos.

2.- Esto permite la reutilización de definiciones abstractas: *messages*, que son descripciones abstractas de los datos intercambiados, y *portTypes*, que son colecciones abstractas de operaciones.

3.- Las especificaciones concretas del protocolo y del formato de datos para un tipo de puerto determinado constituyen un enlace reutilizable.

4.- Un puerto se define por la asociación de una dirección de red y un enlace reutilizable; una colección de puertos define un servicio.

Un contrato de servicio WSDL utiliza los siguientes elementos en su definición:

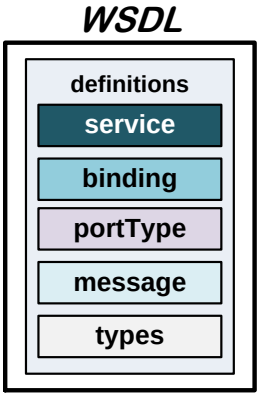


Ilustración 7. Composición de un WSDL

El elemento más externo de un WSDL se llama *definitions*, dado que provee definiciones agrupadas en las siguientes secciones:

- Sección ***types*** (opcional): define los tipos de datos utilizados en los mensajes; de forma predeterminada se utilizan los tipos definidos en la especificación de esquema XML (el elemento *schema* aparece como elemento hijo de *types*). Si se desea utilizar otro sistema de tipos, aparecerá un elemento de extensibilidad que lo identifique bajo el elemento *types*.
- Sección ***message***: proporciona una abstracción común para el paso de mensajes entre el cliente y el servidor, es decir, define los mensajes que implementan el servicio. Los mensajes son construidos en base a los tipos de datos definidos en la sección anterior.
  Normalmente aparecen múltiples elementos *message*, uno para cada mensaje, y cada uno contiene uno o más elementos "*part*" que describen las piezas del contenido del mensaje.

El orden de los mensajes indica el patrón de servicio: Los mensajes *in* son los que llegan al servicio, mientras que los *out* son enviados desde el servicio. Si el orden de los mensajes es *in/out*, entonces el patrón es *request/response*, mientras que si el orden es *out/in* el patrón es *solicit/response*.

- Sección ***portType***: agrupación de operaciones abstractas, donde cada operación tiene uno o más mensajes. Existen cuatro tipos de operaciones:
  - ✓ ***Request-response*** (petición-respuesta): comunicación de tipo *RPC* en la que el cliente realiza una petición y el servidor envía la respuesta.
  - ✓ ***One-way*** (un-sentido): comunicación del estilo *documento* en la que el cliente envía un mensaje pero no recibe respuesta del servidor.
  - ✓ ***Solicit-response*** (solicitud-respuesta): comunicación de tipo *RPC* en la que el servidor envía una petición y el cliente envía de vuelta una respuesta.
  - ✓ ***Notification*** (Notificación): comunicación del estilo *documento* en la que el servidor envía una comunicación al cliente.
- Sección ***binding***: es donde las definiciones WSDL se concretizan, es decir, es una especie de "implementación" de un *portType*, y provee detalles concretos del servicio como:
  - ✓ El protocolo de transporte a usar al enviar y recibir los mensajes: JMS, HTTP o SMTP.
  - ✓ El estilo del servicio: rpc o document.
- Sección ***service***: agrupación de puertos donde la funcionalidad del servicio (el total de sus operaciones) estará disponible. Un puerto es un extremo concreto de un Servicio Web al que se hace referencia por una dirección única**.** En términos técnicos, la sección *service* lista uno o más elementos *port*, donde un *port* consiste en un *portType* junto a su *binding* correspondiente.

Modelar un servicio web consiste en definir cada una de las posibles casuísticas de generación de su contrato de servicio (WSDL): Tipo de datos, composición del elemento *schema*, estilo de servicio (*document* o *rcp*), partes de los mensajes (*part*s), número de mensajes, patrones de intercambio de mensajes (*MEP - Message Exchange Pattern: Request/Response, Solicit/Response,One-Way, Notification*), número de operaciones, protocolo de comunicación (SOAP sobre HTTP, SOAP sobre JMS, etc)…

Los equipos de arquitectura normalmente definen unos patrones de diseño de los servicios con el objetivo de estandarizar el modelado de los mismos. Dichos patrones son específicos para cada organización y supervisar su cumplimiento es una tarea del equipo de gobierno de la misma.

2.5. Conceptualización de la Situación Objetivo

En el alcance de este proyecto se encuentra el diseño e implementación de un *Catálogo de Referencia* para facilitar las tareas de gobierno de una entidad. Adicionalmente, el alcance incluye la elaboración de una herramienta que permita a los analistas mantener dicho Catálogo desde su puesto de usuario y, además, modelar los servicios de la organización.

En última instancia, los objetivos que persigue la implementación de un Catálogo de Referencia para una entidad son: en primer lugar, **ordenar o agrupar los servicios por criterios abstractos de utilidad** que faciliten la búsqueda de los servicios disponibles, y en segundo lugar, permitir una **constante vigilancia y revisión** de los servicios publicados.

Para la consecución de estos objetivos, se plantea el Catálogo de Referencia de una entidad como la integración de los siguientes elementos:

1.- Glosario de términos (campos):

Los **campos** son la unidad básica de información en el catálogo de referencia. Su definición viene dada por su tipología (subconjunto de tipos primitivos) y las restricciones (facets: White space, Max Inclusive, Min Inclusive, Total Digits, Pattern, Enumeration, Max Exclusive, Min Exclusive, Length…) definidas en el estándar de *w3.org* para cada tipo.

2.- Catálogo de Entidades Canónicas:

Las **entidades canónicas** representan las estructuras lógicas de la organización. Se definen a partir de uno o más campos de entre los albergados en el glosario de términos y/o a partir de otras entidades.

Se clasificarán en entidades de *negocio* o *técnicas* dependiendo de su origen. Su representación física consistirá en un esquema XML (XSD) acompañado de meta-información funcional que será de utilidad para los analistas y permitirá la realización búsquedas enriquecidas.

Las entidades deben estar bajo un sistema de control de versiones, que permita mantener el histórico: su composición puede variar por lo que se podrán verse afectados todos los servicios o entidades en los que intervengan si no se mantienen las versiones.

3.- Catálogo de Servicios Web:

Los **WSDL** definen los contratos de servicio web de la entidad. Sus mensajes se componen a partir del catálogo de entidades y de campos simples del glosario (estos últimos pueden ser considerados entidades de un solo elemento).

Para su modelado se ha considerado un diseño basado en la arquitectura de integración de una entidad bancaria:

- Tres capas diferenciadas: channel adapter, business service y wrapper adapter.
- Ubicación de los servicios según modelo BIAN
- Multioperación con MEP *request/response*
- Binding SOAP sobre JMS

NOTA: En cualquier caso el modelado es parametrizable.

Su representación física consistirá en un fichero *WSDL* acompañado de meta-información funcional que será de utilidad para los analistas y permitirá la realización búsquedas enriquecidas.

Además, estarán bajo un sistema de control de versiones, que permita mantener el histórico de versiones: si los campos o entidades que componen una definición de servicio varían, podría verse alterada la funcionalidad del interfaz.

El Catálogo de Referencia estará soportado sobre una solución con una doble tipología de acceso:

1.- Consola de Administración:

Se tratará de la consola nativa del producto que albergue el Catálogo y permitirá a los responsables de la gobernanza SOA realizar tareas de administración y mantenimiento, como por ejemplo: restricción/asignación de permisos sobre objetos y usuarios, evolución del modelo, restauración de configuraciones ante anomalías…

2.- IDE Utils:

Herramienta de consulta y definición de campos, entidades y contratos WSDL. Estará instalada en el equipo del usuario, incluida en el entorno Eclipse de desarrollo, y se conectará con el Catálogo a través de una API de comunicación.

Esta utilidad permitirá al usuario el modelado de entidades y servicios en base a los patrones marcados por el equipo de arquitectura de la organización y, por tanto, evitando múltiples casos de error.

Además, dicha herramienta debe ser parametrizable y permitir una fácil adaptación a los cambios del modelo: nomenclatura, tipologías de campos, topología de servicios, patrones de diseño de contratos, etc.

El siguiente dibujo expone, de modo simplificado, la solución conceptual que se describe en los siguientes capítulos de este documento:

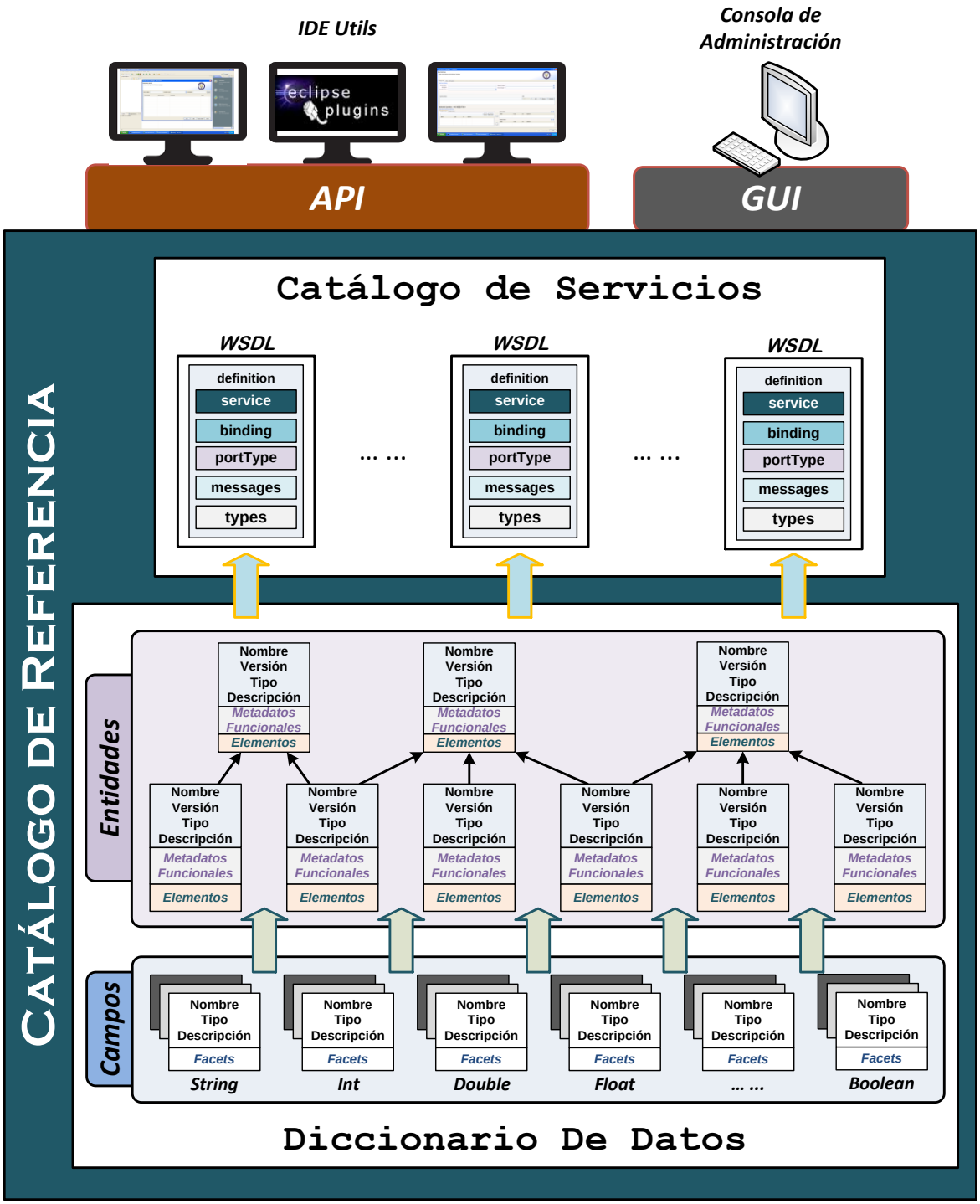


Ilustración 8. Modelo Conceptual de la Solución

Capítulo 3

Catálogo de Referencia

Desde un punto de vista funcional, el Catalogo de Referencia debe ser considerado como un repositorio de recursos centralizado que permite registrar y compartir con todas las áreas de una organización componentes lógicos de diversas tipologías (Campos, Entidades, Contratos de Servicios Web…).

El Catálogo de Referencia propuesto en este proyecto estará constituido por:

- **Diccionario de Datos** (glosario de términos y entidades): almacena todos los elementos, técnicos o funcionales, que forman parte del flujo de datos de un sistema; los equipos responsables del mantenimiento del modelo de datos administrarán y mantendrán dicho diccionario de acuerdo a las implementaciones presentes en las bases de datos de la organización.
- **Catálogo de Servicios Web**: incluye las interfaces WSDL de todos los servicios web publicados por las diferentes áreas. Sus mensajes de entrada/salida estarán definidos a partir del diccionario de datos de la organización y su modelado se regirá por una serie de patrones previamente definidos.

Adicionalmente, todos los elementos registrados pueden ir acompañados con un conjunto de meta datos funcionales para facilitar las tareas de gobierno de los mismos.

En las siguientes secciones se detallará el trabajo realizado para aterrizar el modelo teórico de la solución sobre una plataforma real que soporte el Catálogo de Referencia, dando respuesta a los siguientes requerimientos:

- Recursos a medida
- Modelo de relaciones entre elementos
- Versionado de entidades lógicas y contratos de servicio
- Ordenación de los elementos en base a jerarquías previamente definidas
- Explotación de la información desde sistemas externos
- Capa de presentación amigable

3.1. Selección de la Tecnología

Para la llevar a cabo el desarrollo del Catálogo de Referencia de una organización existen dos posibles caminos a la hora de seleccionar una tecnología de implementación: el primero consiste en la implementación de una **solución a medida**, mientras que el segundo se basa en la utilización de **producto de mercado** con cierto grado de personalización.

La implementación de una solución a medida, además de suponer un esfuerzo adicional de desarrollo, quedaría limitada a la consecución de los requerimientos iniciales. Esto conlleva las siguientes implicaciones:

- Modelo no exportable a otras organizaciones
- Sin capacidades nativas de integración
- La escalabilidad de la solución no está garantizada
- Difícilmente extensible en lo que a su funcionalidad se refiere

Por todo esto, la opción de construir una solución totalmente a medida queda descartada.

Por otro lado, para evaluar los diferentes productos de mercado que podrían dar soporte a un Catálogo de Referencia como el definido dentro de la solución de este proyecto, se ha recurrido a las evaluaciones de Gartner (entidad especializada en la evaluación de productos).

Ilustración 9. Magic Quadrant for On-Premise Application Integration

Entre los diferentes productos de mercado, la mayoría proponen una solución extremadamente asociada a la utilización de la *suite* a que pertenecen y muy poco adaptable y, aunque podrían soportar el Catálogo de Servicios de una organización, no permiten la centralización del Diccionario de Datos (glosario de datos y entidades).

Ilustración 10. Magic Quadrant for SOA Governance

Si bien es cierto que, según el *cuadrante mágico de Gartner*, la *Suite WSO2* no tiene una valoración destacada como integrador de aplicaciones; sin embargo, lo referencia como un proveedor de tecnologías de gobierno SOA histórico, con un alto nivel de madurez y con las siguientes fortalezas:

- Solución de código abierto, muy adecuado para grupos empresariales y proveedores de software independientes.
- Tecnología consistente y multiusuario de modo nativo.
- Modelo de negocio ya probado, con buen soporte empresarial

Resumen:

La herramienta seleccionada para soportar el **Catálogo de Referencia** es la aplicación de libre distribución **WSO2 Governance Registry** en su modalidad Carbon (instalación *On-Premise*), la cual da cobertura a todos los requerimientos del modelo propuesto y proporciona amplias capacidades de integración a través de APIs:

Registry API

The **Registry API** describes how developers can use WSO2 Governance Registry in their developments. Here you can find the detailed information about basic Registry functions, usage of the extending APIs provided by the Registry, about OSGi Support, custom Query and lots of other helpful features to manage Governance Registry using API.

Please, visit the following pages for more information about the Registry API:

- Basic Registry API Knowledge — General information about the Registry API and its basic functions.
- Custom Query — General information about Custom Queries.
- OSGi Support — General information about the OSGi support in the WSO2 Governance Registry.
- Custom User Interface — Information about custom user interface.
- Web Services API — General Information about the WEB Services API in Governance Registry.
- Governance API — General information about the Governance API.
- JCR API — General information about the JCR API.
- The RegistryContext Class — Description of the Registry Context class and its attributes.
- The RegistryClientUtils Class — Description of the RegistryClientUtils class and its methods.
- Java Documentation — The list of helpful Java docs.
- Accessing Registry Remotely through API — Instructions on how to manage Remote Registry with API in Governance Registry.
- API-Level Access to the Subscription Manager — Description of the API-level access to the subscription manager in the Governance Registry.
- Access Governance Registry through Javascript — General information about the javascript API for accessing the WSO2 Governance Registry.
- Registry REST API

3.2. Modelo Conceptual de la Solución

Cada uno de los elementos almacenados en el Catálogo de Referencia vendrá representado por uno o más recursos en la herramienta de registro (WSO2 Governance Registry).

El registro debe servir de soporte para catalogar los siguientes elementos lógicos y sus relaciones: Campos, Entidades y Contratos de Servicio (WSDL). En la siguiente imagen pueden visualizarse los recursos que componen cada elemento del Catálogo de Referencia:

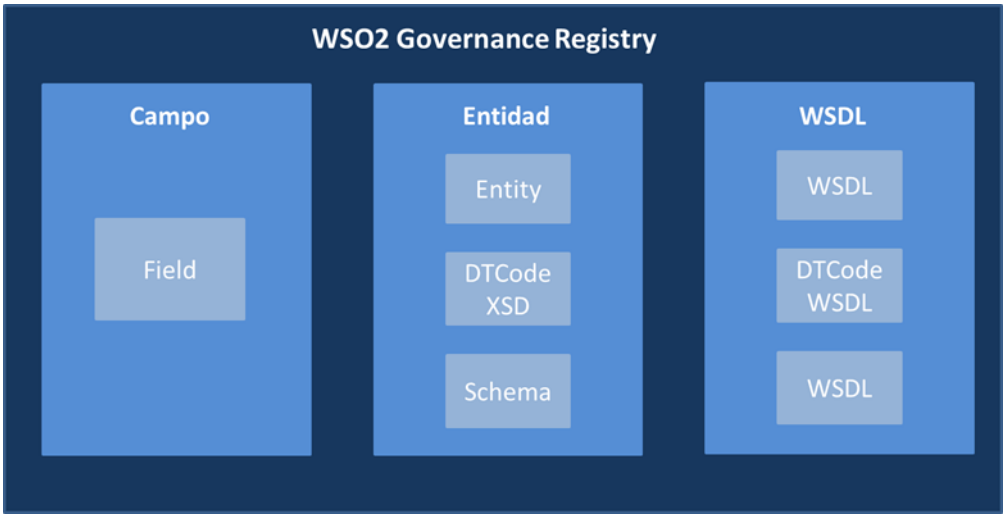


Ilustración 11. Modelo conceptual del Catálogo de Referencia

Campo: Es la unidad básica del diccionario de datos y está representado en WSO2 Governance Registry por el recurso Field definido para tal efecto.

Entidad: Es una agrupación de Campos y/o Entidades que representan una unidad lógica del modelo canónico, por ejemplo, los datos de una persona. El elemento Entidad está compuesto por los siguientes recursos del WSO2 Governance Registry:

- Entity: metadatos técnicos y funcionales
- DTCodeXSD: serialización JSON de la entidad
- Schema: contenerdor XSD de la entidad

Contrato de Servicio (WSDL): Es la interfaz pública que describe las operaciones del servicio. El elemento Contrato de Servicio está compuesto por los siguientes recursos del WSO2 Governance Registry:

- Servicio: metadatos técnicos y funcionales
- DTCodeWSDL: serialización JSON del contrato
- Schema: contenerdor WSDL del contrato

3.3. Blueprint de Arquitectura

A continuación, se muestra el diseño a alto nivel de los componentes arquitecturales de la solución:

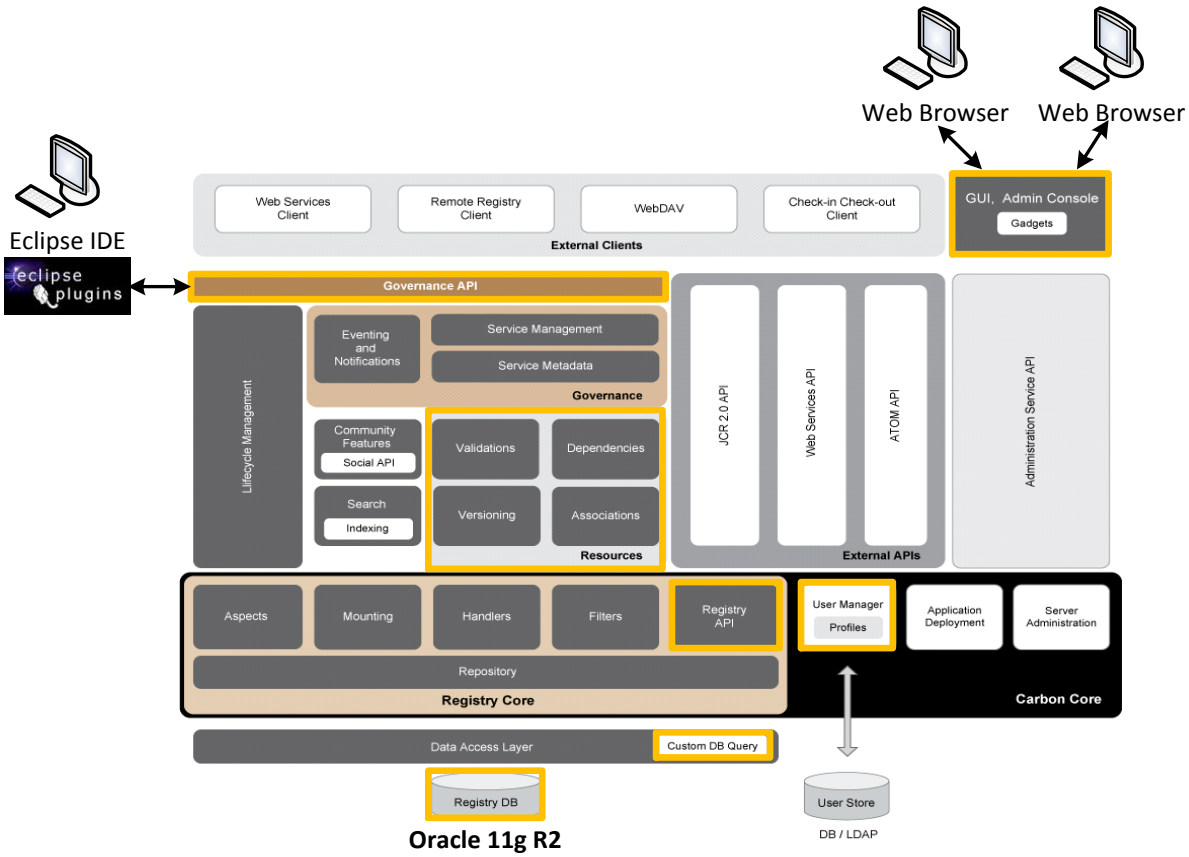


Ilustración 12. Blueprint de Solución

NOTA: Cada una de las piezas arquitecturales resaltadas, están de una u otra forma involucradas en el diseño de la solución final:

- Se han añadido artefactos de visualización en la Consola de Administración (*Field*/*Entity*)
- Se han establecido relaciones entre recursos, se ha ampliado la capacidad de versionado (solución no nativa) y se han eliminado las validaciones de producto.
- Se ha hecho uso de las capacidades de integración a través de las piezas Governance API y Registry API
- La capa de datos ha sido extendida a través de consultas a medida (en adelante, *custom queries*) y con la externalización de la base de datos: Oracle 11g R2 Express Edition

3.4. Extensión de funcionalidad

Esta sección pretende describir las *ampliaciones de funcionalidad* realizadas sobre WSO2 Governance Registry para permitir su uso como Catálogo de Referencia, así como su explotación desde la herramienta del entorno local de desarrollo Eclipse que se presenta en el próximo capítulo (*IDE Utils*).

3.4.1. Ficheros de Configuración

En la siguiente tabla se listan todos los ficheros de configuración que han requerido cambios, junto con una descripción a alto nivel de los mismos y su ruta relativa:

Fichero	Descripción	Cambios	Ruta
registry.xml	Fichero que contiene propiedades y handlers de WSO2 Governance Registry.	• Los paths de almacenamiento se han cambiado para que apunten a la ruta relativa /uc3m. • Los handlers de validación internos del Registry se han deshabilitado. • El handler de creación automática de servicios se ha deshabilitado.	/repository/conf/
master-datasources.xml	Fichero con la configuración de la conexión de la Base de Datos	• Se han configurado los datasources WSO2_CARBON_DB y WSO2AM_DB para conectar con una BD Oracle	/repository/conf/datasources/
web.xml	Fichero que describe cómo desplegar la aplicación web *Carbon Bridge*.	• Timeout de sesión ampliado.	/repository/conf/tomcat /carbon/WEB-INF/
identity.xml	Fichero que se utiliza para la autenticación en WSO2 Governance Registry.	• Modificada la generación de token para la autenticación.	/repository/conf/
catalina-server.xml	Fichero con la configuración del servidor Tomcat.	• Modificadas las opciones de seguridad de la conexión.	/repository/conf/tomcat/

Tabla 1. Fichero de configuracion de WSO2 Governance Registry

3.4.2. Recursos del Catálogo

La principal personalización del Catálogo de Referencia es la relativa a la definición de los recursos que almacenará. A continuación, se describe la implementación técnica de los artefactos registrados en la herramienta para cada una de las únidades funcionales: Campo, Entidad y Contrato de Servicio Web

3.4.2.1. Campo

Field es un artefacto totalmente *a medida*, con identificador de tipología (*mediatype*) no estándar definido en el ámbito de este proyecto: **application/vnd.wso2-field+xml.**

En la siguiente tabla se describen los atributos con los que se ha definido el artefacto:

Atributo	Descripción
overview_name	Nombre del Campo
overview_type	Tipo del Campo
overview_description	Descripción del Campo
facets_length	Longitud del Campo
facets_maxLength	Longitud máxima del Campo
facets_minLength	Longitud mínima del Campo
facets_totalDigits	Número total de dígitos del Campo
facets_fractionDigits	Especifica el número de decimales.
facets_pattern	Especifica la secuencia de caracteres aceptados.
facets_enumeration	Indica si el Campo es parte de una enumeración.
facets_whiteSpace	Indica el comportamiento frente a los espacios en blanco.
facets_maxInclusive	Especifica el número máximo. Para un valor numérico, debe ser igual o menor que el valor especificado.

facets_maxExclusive	Especifica el número máximo. Para un valor numérico, debe ser menor que el valor especificado.
facets_minInclusive	Especifica el número mínimo. Para un valor numérico, debe ser igual o mayor que el valor especificado.
facets_minExclusive	Especifica el número mínimo. Para un valor numérico, debe ser mayor que el valor especificado.
attributes_status	Especifica si el Campo está bloqueado.

Tabla 2. Atributos del recurso campo

NOTA: Los atributos que empiezan por **facets_** corresponden a restricciones de los valores de los Campos, y aplicaran según su tipología, de acuerdo al estándar de w3.org

3.4.2.2. Entidad

1.-Recurso Entity

Entity es un artefacto totalmente *custom*, con *mediatype* no estándar **application/vnd.wso2-entity+xml**.

En la siguiente tabla se describen los atributos que componen el artefacto:

Atributo	Descripción
overview_name	Nombre de la Entidad
overview_version	Versión de la Entidad
overview_type	Tipo de la Entidad
overview_description	Descripción de la Entidad
storage_schema	Ruta de almacenamiento de XSD
metadata_name	Nombre del metadato funcional
metadata_value	Valor del metadato.
attributes_dtcode	Valor del metadato.
attributes_status	Especifica si la Entidad está bloqueada o desbloqueada.
attributes_infoComment	Comentario de la versión del WSDL.

attributes_infoUser	Usuario que ha cargado la versión del WSDL.

Tabla 3. Atributos del recurso entidad

2.- Recurso DTCodeXSD

El recurso DTCodeXSD es un recurso genérico, con *mediatype* **Unknown**, definido para almacenar la serialización JSON del elemento Entidad del Catálogo de Referencia. En la siguiente imagen se muestra la estructura del contenido del DTCodeXSD:

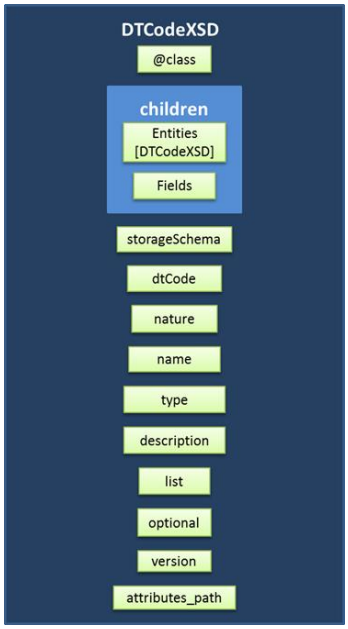


Ilustración 13.Estructura JSON para almacenamiento de entidades

3.- Recurso Schema

El recurso Schema, con *mediatype* **Schema**, contiene la definición XSD correspondiente al elemento Entidad.

Ejemplo:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeFormDefault="unqualified" elementFormDefault="unqualified">
<xs:complexType name="TestNewVersionPlugin">
<xs:sequence>
<xs:element name="key">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:whiteSpace value="preserve"/>
<xs:enumeration value="true"/>
<xs:minLength value="20"/>
<xs:maxLength value="30"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="value">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:whiteSpace value="preserve"/>
<xs:enumeration value="false"/>
<xs:length value="0"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:schema>
```

3.4.2.3. Contrato de Servicio Web

1.- Recurso WSDL

WSDL es un artefacto nativo de Gobvernance Registry, pero ha sido modificado para que albergue toda la información funcional del servicio al que representa el WSDL.

En la siguiente tabla se describen los atributos que definen el Servicio:

Atributo	Descripción
attributes_name	Nombre del WSDL
attributes_dtcode	Ruta de la Serialización JSON del WSDL.
attributes_infoComment	Comentario de la versión del WSDL.
attributes_infoUser	Usuario que ha cargado la versión del WSDL.
attributes_description	Descripción del WSDL.
attributes_interfacetype	Tipo del WSDL.
attributes_servicename	Nombre del servicio del WSDL.
attributes_status	Especifica si el WSDL está bloqueada o desbloqueada.
attributes_user	Usuario que ha creado el WSDL.
attributes_version	Última versión del WSDL cargada.

attributes_wsdlTemplateId	Identificador de la versión de la plantilla que se ha utilizado para generar el WSDL.

Ilustración 14. Atributos del recurso WSDL

2.- Recurso DTCodeWSDL

El recurso DTCodeWSDL es un recurso a medida, con *mediatype* **application/x-json-dtCodeWsdl**, definido para almacenar la serialización JSON de los contratos de servicio del Catálogo de Referencia. En la siguiente imagen se muestra la estructura del contenido del DTCodeWSDL.

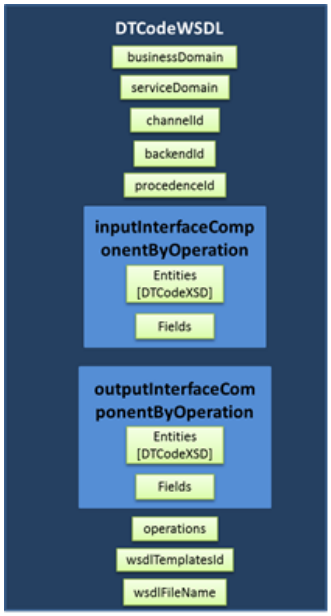


Ilustración 15. Estructura JSON para almacenamiento de WSDL

3.4.3. Versionado de Elementos

WSO2 Governance Registry ofrece capacidades de versionado, pero no es válido para el diseño propuesto por este PFC: se trata de un versionado secuencial, basado en un incremental independiente de la naturaleza del elemento, que no contempla la posibilidad de que un componente de software esté compuesto por mas de un recurso en el registro.

Para dar solución a esta necesidad, se ha diseñado un sistema propio de versiones, basado en las rutas lógicas de almacenamiento de los recursos en el registro.

En la siguiente tabla se especifica el versionado de los recursos de Registry para cada elemento del Catálogo de Referencia.

	Artefacto	DTCode
Campo	Sin versionado	Sin versionado
Entidad	El artefacto Entity, se versiona guardando bajo un directorio (con el nombre de la Entidad) las diferentes versiones de la misma.	DTCodeXSD tiene versionado. Las versiones se almacenan bajo un directorio que lleva el nombre del artefacto Entity, cada subcarpeta con su fichero es una versión diferente.
WSDL	-	DTCodeWSDL tiene versionado. Las versiones se almacenan bajo un directorio que lleva el nombre del artefacto WSDL, cada subcarpeta con su fichero es una versión diferente.

Tabla 4. Relación de elementos versionados en el registro

En la siguiente imagen puede verse un ejemplo de versionado de un elemento de tipo Entidad a traves de la consola de WSO2 Governance Registry:

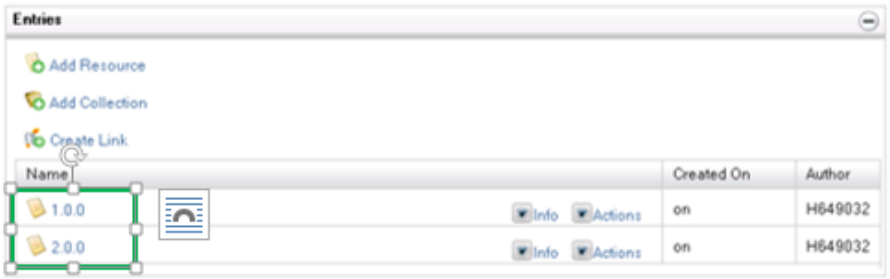


Ilustración 16. Versionado de elementos en el Catálogo de Referencia

3.4.4. Definición de Custom Queries

Las *custom queries* son recursos genéricos que contienen consultas de la información de los elementos del Catálogo de Referencia. Las *queries* se ejecutan contra la Base de Datos de WSO2 Governance Registry a traves de su capa nativa de acceso a datos.

Estas consultas son consideradas por el registry como tales por la tipología del recurso que las almacena (*mediatype:* ***application/vnd.sql.query***) y por la ruta lógica en la que está almacenado dicho recurso dentro de WSO2 Governance Registry:

/_system/config/repository/components/org.wso2.carbon.registry/queries

Por ejemplo, una custom query que recupera la información del nombre de un recurso *Field* de Registry sería la siguiente:

```
SELECT REG_PATH_ID, REG_NAME FROM REG_RESOURCE
WHERE REG_MEDIA_TYPE='application/vnd.wso2-field+xml'
AND
UPPER(REG_NAME) LIKE UPPER(?)
```

En tiempo de ejecución los símbolos "**?**" son sustituidos por los valores apropiados. Las *custom queries* creadas para satisfacer las necesidades del Catálogo de Referencia son las descritas en la siguiente tabla:

Custom Query	Descripción
query_importartifact_path	Recupera la ruta en WSO2 Governance Registry de un artefacto.
query_service_name	Recupera información de un recurso de tipo Service de WSO2 Governance Registry a partir del nombre del Service.
query_schema_name	Recupera información de un recurso de tipo Schema de WSO2 Governance Registry a partir del nombre del Schema.
query_wsdl_name_equal	Recupera información de un recurso de tipo WSDL de WSO2 Governance Registry. La condición de la consulta es que el nombre del WSDL debe ser igual al nombre buscado.
query_wsdl_name	Recupera información de un recurso de tipo WSDL de WSO2 Governance Registry. La condición de la consulta es que el nombre del WSDL debe ser parecido al nombre buscado.
query_field_name	Recupera información de un recurso de tipo Field de WSO2 Governance Registry a partir del nombre del Field.
query_field_type	Recupera información de un recurso de tipo Field de WSO2 Governance Registry a partir del tipo del Field.
query_field_description	Recupera información de un recurso de tipo Field de WSO2 Governance Registry a partir de la descripción del Field.
query_entity_name_equal	Recupera información de un recurso de tipo Entity de WSO2 Governance Registry a partir del nombre de la Entity. La condición de la consulta es que el nombre de la Entity debe ser igual al nombre buscado.

query_entity_name	Recupera información de un recurso de tipo Entity de WSO2 Governance Registry a partir del nombre de la Entity. La condición de la consulta es que el nombre de la Entity debe ser parecido al nombre buscado.
query_entity_type	Recupera información de un recurso de tipo Entity de WSO2 Governance Registry a partir del tipo de la Entity.
query_entity_description	Recupera información de un recurso de tipo Entity de WSO2 Governance Registry a partir de la descripción de la Entity.
query_field_nametype_equal	Recupera información de un recurso de tipo Field de WSO2 Governance Registry a partir del nombre y el tipo del Field.
query_wsdl_multiple_all_versions	Recupera información de los recursos de tipo WSDL de WSO2 Governance Registry que su versión esté en el intervalo de versiones de la consulta.
query_field_multiple_typefix	Recupera información de los recursos de tipo Field de WSO2 Governance Registry a partir del nombre, tipo y la descripción de la consulta.
query_field_multiple	Recupera información de los recursos de tipo Field de WSO2 Governance Registry a partir del nombre, tipo y la descripción de la consulta.
query_field_name_history	Recupera información histórica de un recurso de tipo Field de WSO2 Governance Registry a partir del nombre, tipo y descripción del Field.
query_entity_multiple	Recupera información de los recursos de tipo Entity de WSO2 Governance Registry a partir del nombre, tipo y descripción de la consulta.
query_entity_multiple_last_version	Recupera información la última versión de los recursos de tipo Entity de WSO2 Governance Registry a partir del nombre, tipo y descripción de la consulta.
query_wsdl_multiple	Recupera información de los recursos de tipo WSDL de WSO2 Governance Registry a partir del nombre, tipo y descripción de la consulta.

Tabla 5. Custom Queries

3.4.5. Sistema de Reserva de Elementos: Check-Out

La herramienta de registro no comtempla de forma nativa, la reserva o bloqueo de recursos. Sin embargo, el bloqueo de los elementos del Catálogo de Referencia es necesario para evitar que más de un desarrollador modifique el mismo elemento a la vez y se pierdan los cambios realizados.

Para ofrecer la funcionalidad de reserva de los elementos del Catálogo de Referencia, se ha definido un atributo llamado "*status*" en los artefactos Field, Entity y WSDL. Dicho atributo cambiará su valor en funcion de si esta siendo editado por un usuario o no: *locked* (bloqueado) o *unlocked* (desbloqueado).

En la siguiente imagen puede verse un ejemplo de un recurso WSDL en estado desbloqueado:

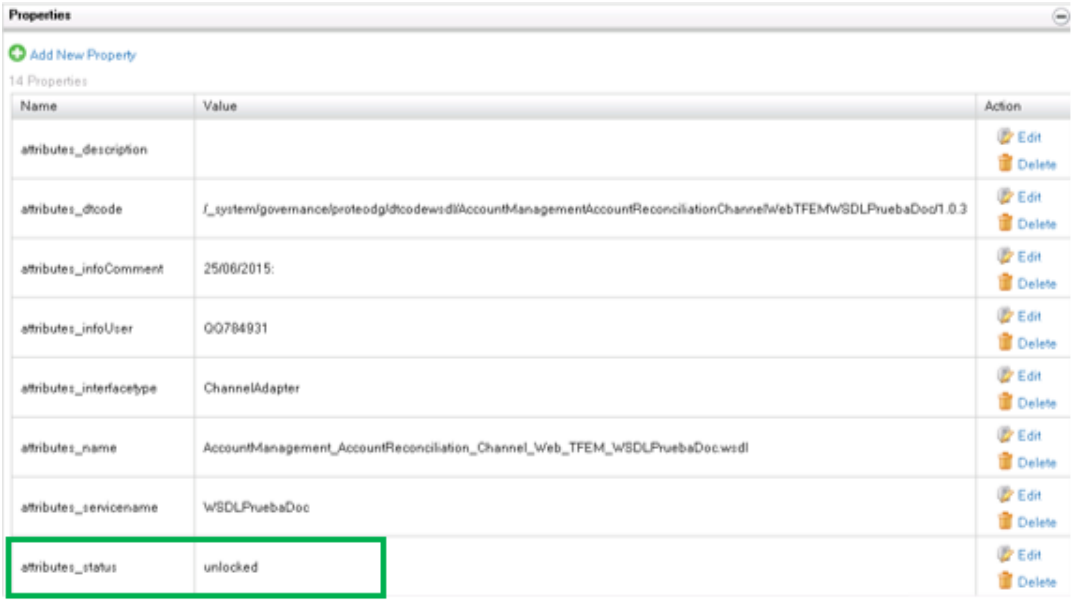


Ilustración 17. Sistema de reserva de elementos

Una vez bloqueado un elemento del registro, solo podrá ser desbloqueado por el usuario que lo bloqueo y, en situaciones anómalas, a traves de la consola de WSO2 Governance Registry con el usuario administrador.

3.5. Almacenamiento Lógico y Capa de Presentación

La herramienta de registro, *WSO2 Governance Registry*, desde un punto de vista técnico, es un contenedor de recursos con una serie de capacidades adicionales, entre las cuales destacan las provistas por su capa de visualización (*GUI*).

Los recursos son almacenados sobre el modelo de datos de la herramienta, pero son identificados a través de un modelo lógico de rutas; es decir, los recursos se almacenan sobre una serie de tablas fijas, gestionadas por el núcleo de *Registry*, pero son identificadas y mostradas al usuario como si de una estructura de directorios se tratase.

La raíz de almacenamiento lógico viene dada por la configuración del WSO2 Governance Registry; en nuestro caso, es:

/_system/governance/uc3m/

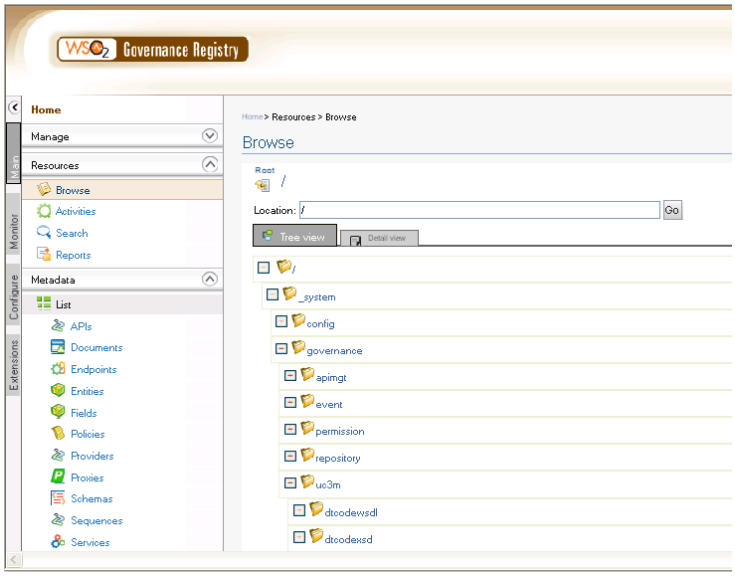


Ilustración 18. Visualización de recursos a través del Browse

Para cada uno de los recursos almacenados en el Catálogo, se ha definido un modelo de almacenamiento alineado con sus necesidades: clasificación, versionado…

A continuación, expondremos el modelo de la estructura de almacenamiento lógico, incluyendo un ejemplo de la visualización a través de la consola de gestión, para cada uno de los artefactos que componen el Catálogo de Referencia:

Campos:

fields/[Tipo]/[Nombre]

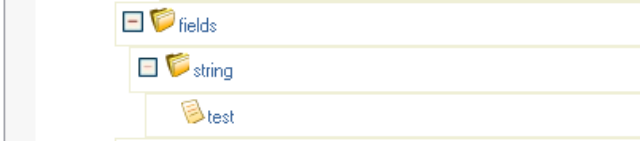


Ilustración 19. Ejemplo de almacenamiento field

Entidades:

entities/[Tipo]/[Nombre]/[Versión]

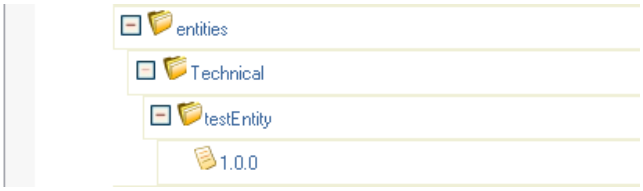


Ilustración 20. Ejemplo de almacenamiento entity

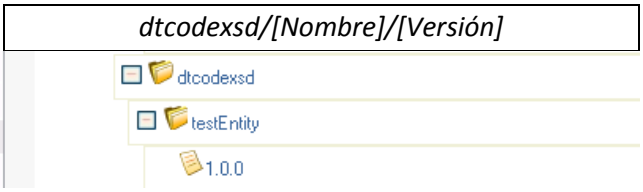


Ilustración 21.Ejemplo de almacenamiento dtcodexsd

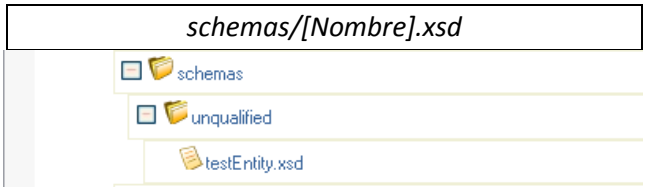


Ilustración 22. Ejemplo de almacenamiento xsd

Contratos de Servicio:

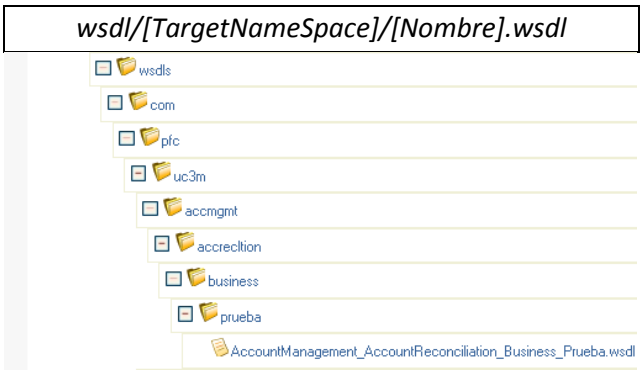


Ilustración 23. Ejemplo de almacenamiento wsdl

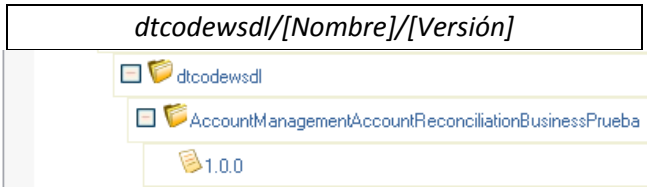


Ilustración 24. Ejemplo de almacenamiento dtcodewsdl

La consola de administración también permite la inclusión de ventanas de visualización de los artefactos a través de la implementación de configuraciones directamente incluidas como extensiones de tipos de artefacto en formato xml (***Extensions > Artifact Types***):

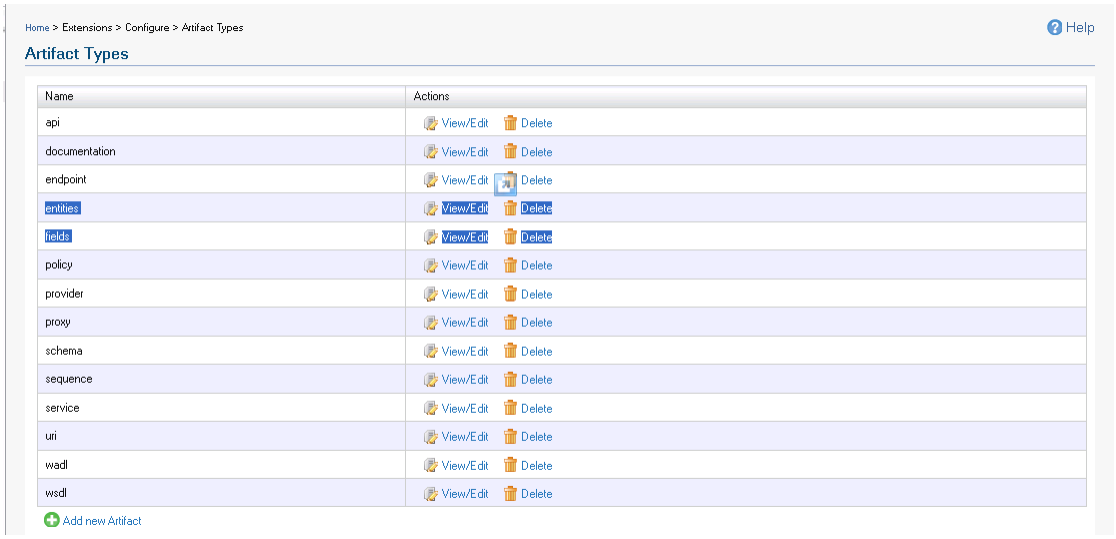


Ilustración 25. Listado de artefactos gráficos

La visualización de dichos artefactos en modo gráfico se muestra en las siguientes ilustraciones:

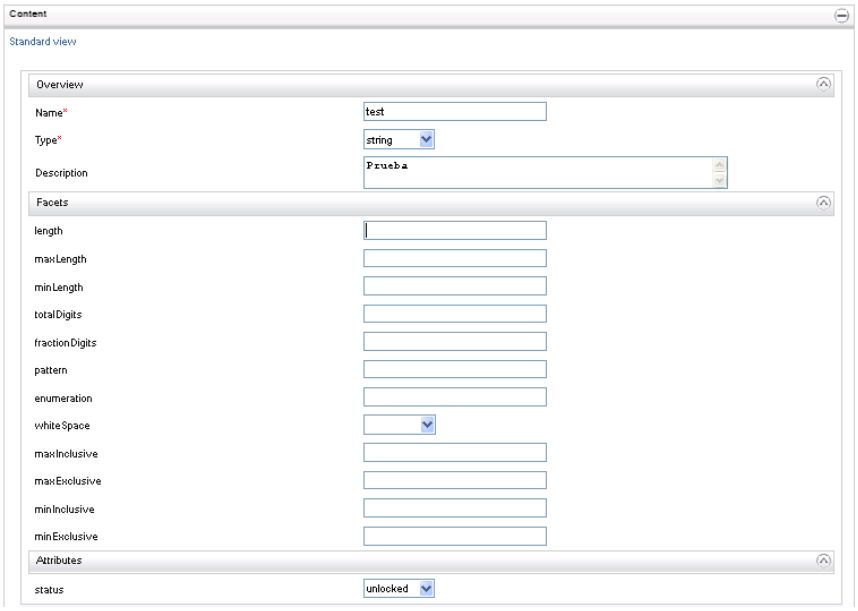


Ilustración 26. Artefacto gráfico de consola Field

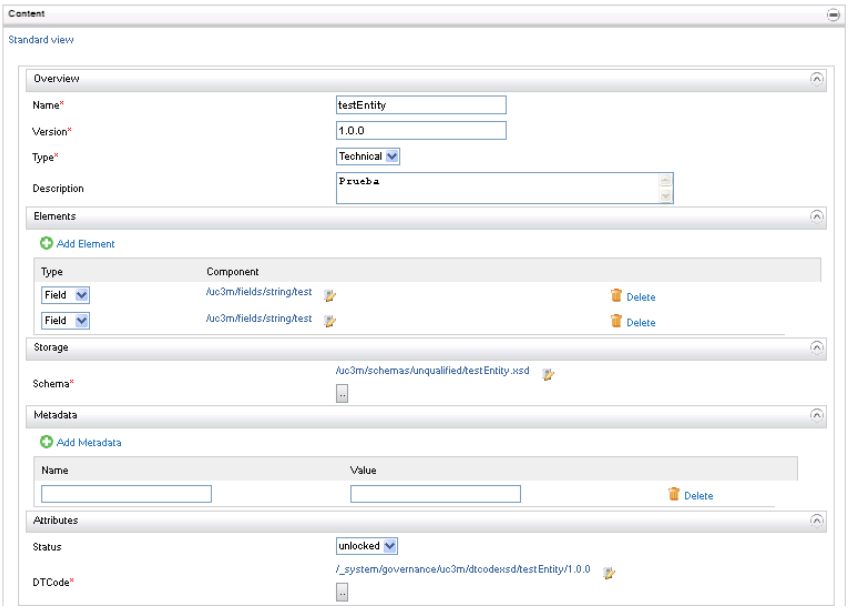


Ilustración 27. Artefacto gráfico de consola Entity

Capítulo 4

Herramienta IDE Utils

"Si tu única herramienta es un martillo,
tiendes a tratar cada problema como si fuera un clavo"

Abraham Maslow.

Eclipse es el *framework* de desarrollo más extendido entre los desarrolladores de sistemas distribuidos. Al tratarse de una solución de código abierto basada en tecnología *Java* permite un alto grado de personalización a través de extensiones del producto, denominadas *plug-ins*.

Si bien es cierto que los Entornos de Desarrollo Integrados (en inglés, *IDE)* de los diferentes proveedores cubren cada vez un abanico mayor de tecnologías y permiten extensiones de la funcionalidad nativa, en ningún caso se trata de soluciones tan extendidas y diversificadas como Eclipse.

Además, en los últimos años, ámbitos en los que tradicionalmente se usaban tecnologías nativas de desarrollo, como mainframe o iSeries, cada vez está más extendido el uso de *IDEs* basados en Eclipse.

Por todo esto, la herramienta *IDE Utils* ha sido definida como una extensión de la funcionalidad del entorno Eclipse, esto es como *plug-in Eclipse* implementado en *Java*.

El objetivo es ofrecer a los usuarios de la organización una herramienta de consulta y gestión del Catálogo de Referencia, que sea intuitiva, amigable y esté integrada con su entorno cotidiano de desarrollo. Las funcionalidades que implementa la utilidad son las siguientes:

- Alta, búsqueda y modificación de Campos en el Catálogo de Referencia.
- Creación, búsqueda, modificación y versionado de Entidades.
- Modelado, búsqueda, modificación y versionado de Contratos de Servcio (WSDL).

En este capítulo se hace foco en el planteamiento técnico de la solución que da respuesta a los objetivos anteriormente mencionados, esto es, una descripción técnica de la implementación a alto nivel.

4.1. Modelo de Conocimiento

A continuación, se muestra la estructura del proyecto de la herramienta IDE Utils (*Eclipse plug-in*):

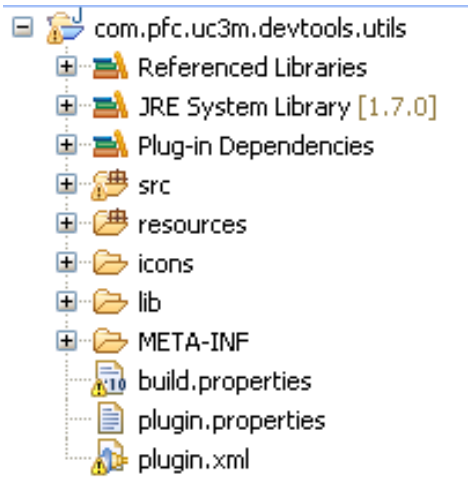


Ilustración 28. Estructura de proyecto IDE Utils

Los directorios mostrados en la imagen contienen el código y los recursos necesarios para el funcionamiento del plug-in. En la siguiente tabla se describe el contenido de los directorios anteriores:

Directorio	Descripción
src	Directorio que contiene la lógica del plug-in dividida en veinticinco paqueterías diferentes. - com.pfc.uc3m.devtools.utils.handlers contiene las clases con los handlers que ejecutan las ventanas de gestión de artefactos de tipo Entidad, Campo e Interfaz. - com.pfc.uc3m.devtools.utils.interfacegenerator contiene las clases *action delegate* del plug-in que ejecutan las ventanas de gestión de entidades, interfaces y campos. - com.pfc.uc3m.devtools.utils.interfacegenerator.[entities|fields|interfaces].dialogs estos tres paquetes contienen las clases de la interfaz gráfica de las opciones de gestión de entidades, campos e interfaces. - com.pfc.uc3m.devtools.utils.interfacegenerator.serialization contiene las clases de serialización a base 64. - com.pfc.uc3m.devtools.utils.interfacegenerator.environment contiene las clases de constantes usadas por el plug-in. - com.pfc.uc3m.devtools.utils.interfacegenerator.utils contiene las clases con funciones de utilidades de apoyo, utilizadas por el resto de clases del plug-in.

	- com.pfc.uc3m.devtools.utils.interfacegenerator.versionableObjects contiene las clases que representan los objetos de versionado de Campo, Entidad, Interfaz del Catálogo de Referencia y componentes del artefacto Interfaz. - com.pfc.uc3m.devtools.utils.interfacegenerator.versionableObjects.entityVersions contiene una clase para el versionado del artefacto Entidad del Catálogo de Referencia. - com.pfc.uc3m.devtools.utils.interfacegenerator.versionableObjects.fieldVersions contiene una clase para el versionado del artefacto Campo del Catálogo de Referencia. - com.pfc.uc3m.devtools.utils.interfacegenerator.versionableObjects.interfaceComponentVersions contiene una clase para el versionado de componentes del artefacto Interfaz. - com.pfc.uc3m.devtools.utils.interfacegenerator.versionableObjects.interfaceDefinitionVersions contiene una clase para el versionado del artefacto Interfaz del Catálogo de Referencia. - com.pfc.uc3m.devtools.utils.interfacegenerator.visualUtils contiene las clases con funciones de utilidad de las interfaces visuales. - com.pfc.uc3m.devtools.utils.proxyapigovernance contiene las clases implementadoras que interactúan con los artefactos del Catálogo de Referencia. - com.pfc.uc3m.devtools.utils.proxyapigovernance.commons contiene una clase que define el status (*locked* o *unlocked*) del artefacto del Catálogo de Referencia. - com.pfc.uc3m.devtools.utils.proxyapigovernance.domain.[entity|fields|interfaces|metadata] estos cuatro paquetes contienen las clases que definen las propiedades y los metadatos de los artefactos Entidad, Campo, Interfaz. - com.pfc.uc3m.devtools.utils.proxyapigovernance.impl contiene una clase para indicar si se utilizará la implementación real de la API o una mock. - com.pfc.uc3m.devtools.utils.proxyapigovernance.impl.accenture contiene las clases de implementación de la API que permite crear, guardar y buscar artefactos de tipo Campo, Entidad, Interfaz y Schema del Catálogo de Referencia. - com.pfc.uc3m.devtools.utils.proxyapigovernance.impl.mock contiene las clases mock de la implementación de la API. - com.pfc.uc3m.devtools.utils.proxyapigovernance.impl.utils contiene una clase de utilidades para la conversión de propiedades a propiedades de cada tipo (Interfaz, Campo y Entidad) y conversión inversa. - com.pfc.uc3m.devtools.utils.proxyapigovernance.impl.protocols contiene las clases *interface* de la implementación de la API.

resources	Directorio que contiene los recursos, necesarios por el plug-in, divididos en tres subdirectorios. - **properties** contiene los ficheros de propiedades que se recuperan desde las clases del código. Estos ficheros incluyen propiedades de reglas de nomenclatura, propiedades de los tipos de artefactos (Entidad, Campo e Interfaz) y metadatos de los tipos Entidad e Interfaz del Catálogo de Referencia. - **templates** contiene un icono con el logo de UC3M y una plantilla del artefacto Entidad. - **templates.wsdl** contiene las plantillas de los WSDLs de las capas Business, Channel y Wrapper. Además hay plantillas de *bindings*, *portype*, *message* y *schemas* de un WSDL. Contiene un fichero de propiedades que se utilizan al cargar el WSDL de un servicio al Catálogo de Referencia.
icons	Directorio que contiene los ficheros de imágenes de los iconos del menú contextual de Eclipse para gestionar las entidades, campos e interfaces.
libs	Directorio que contiene las librerías utilizadas por el plug-in.

Tabla 6. Modelo de Conocimiento de IDE Utils

4.2. Descripción de Módulos Principales

La herramienta IDE Utils, tiene como principales módulos las clases Java que contienen la lógica de la aplicación que permiten crear, editar y exportar Campos, Entidades y Contratos de Servicio del Catálogo de Referencia.

Adicionalmente, se incluye como módulo principal una librería que aglutina la integración con WSO2 Governance Registry, ***GovernanceAPI.jar***, que por estrategia de diseño fue implementada como un elemento aislado para poder ser reaprovechada en futuras aplicaciones.

En la siguiente tabla se describen los módulos principales y se detallan las funcionalidades más importantes de cada uno.

Módulo	Descripción y Funcionalidaes
EntitiesHandler.java	Esta clase contiene una función para la ejecución de la ventana de gestión de artefactos Entidad del Catálogo de Referencia. ▪ Ejecutar la ventana de gestión de artefactos Entidad.
FieldsHandler.java	Esta clase contiene una función para la ejecución de la ventana de gestión de artefactos Campo del Catálogo de Referencia. ▪ Ejecutar la ventana de gestión de artefactos Campo.

InterfacesHandler.java	Esta clase contiene una función para la ejecución de la ventana de gestión de artefactos Interfaz del Catálogo de Referencia. ▪ Ejecutar la ventana de gestión de artefactos Interfaz.
NewEntityDialog.java	Esta clase contiene la estructura de la ventana de creación y edición de una Entidad y las funciones necesarias para crear, editar y exportar Entidades. ▪ Creación de una Entidad en el Catálogo de Referencia. ▪ Recuperación de versiones anteriores de Entidades. ▪ Carga del historial de versiones de una Entidad. ▪ Búsqueda de Campos. ▪ Búsqueda de Entidades. ▪ Creación de versión de una Entidad. ▪ Validación de una Entidad. ▪ Creación de la ventana de confirmación de creación/edición de Entidad. ▪ Generación de fichero XSD de una Entidad. ▪ Carga de los metadatos de una Entidad. ▪ Eventos de los componentes de la interfaz gráfica de creación/edición de Entidades.
EntitiesDialog.java	Esta clase contiene la estructura de la ventana de gestión de Entidades y las funciones necesarias para buscar y cargar Entidades del Catálogo de Referencia. ▪ Eventos de los componentes de la interfaz gráfica de gestión de Entidades. ▪ Búsqueda de Entidades del Catálogo de Referencia. ▪ Exportación de una Entidad del Catálogo de Referencia.
NewFieldDialog.java	Esta clase contiene la estructura de la ventana de creación y edición de un Campo y las funciones necesarias para crear y editar Campos. ▪ Eventos de los componentes de la interfaz gráfica de creación/edición de Campos. ▪ Creación de un Campo en el Catálogo de Referencia. ▪ Carga de metadatos de un Campo.
FieldsDialog.java	Esta clase contiene la estructura de la ventana de gestión de Campos y las funciones necesarias para buscar y cargar Campos del Catálogo de Referencia. ▪ Eventos de los componentes de la interfaz gráfica de gestión de Campos. ▪ Búsqueda de Campos del Catálogo de Referencia.
NewInterfaceDialog.java	Esta clase contiene la estructura de la ventana de creación y edición de una Interfaz y las funciones necesarias para crear, editar y exportar Interfaces.

	▪ Creación de una Interfaz en el Catálogo de Referencia. ▪ Recuperación de versiones anteriores de Interfaces. ▪ Carga del historial de versiones de una Interfaz. ▪ Búsqueda de Campos. ▪ Búsqueda de Entidades. ▪ Creación de versión de una Interfaz. ▪ Validación de una Interfaz. ▪ Creación de la ventana de confirmación de creación/edición de Interfaz. ▪ Generación de fichero WSDL de una Interfaz. ▪ Carga de los metadatos de una Interfaz. ▪ Eventos de los componentes de la interfaz gráfica de creación/edición de Interfaces.
InterfacesDialog.java	Esta clase contiene la estructura de la ventana de gestión de Interfaces y las funciones necesarias para buscar y cargar Interfaces del Catálogo de Referencia. ▪ Eventos de los componentes de la interfaz gráfica de gestión de Interfaces. ▪ Búsqueda de Interfaces del Catálogo de Referencia. ▪ Exportación de una Interfaz del Catálogo de Referencia.
Entity.java	Esta clase contiene funciones para definir y recuperar información del objeto de versionado Entidad. ▪ Definición de la información del objeto de versionado Entidad. ▪ Recuperación de la información del objeto de versionado Entidad.
Field.java	Esta clase contiene funciones para definir y recuperar información del objeto de versionado Campo. ▪ Definición de la información del objeto de versionado Campo. ▪ Recuperación de la información del objeto de versionado Campo.
InterfaceComponent.java	Esta clase contiene funciones para definir y recuperar información del objeto de versionado de componente de la Interfaz. ▪ Definición de la información del objeto de versionado de componente de la Interfaz. ▪ Recuperación de la información del objeto de versionado de componente de la Interfaz.
InterfaceDefiniton.java	Esta clase contiene funciones para definir y recuperar información del objeto de versionado Interfaz. ▪ Definición de la información del objeto de versionado Interfaz. ▪ Recuperación de la información del objeto de versionado Interfaz.
EntityProperties.java	Esta clase contiene funciones para insertar y recuperar propiedades y metadatos de los artefactos Entidad, Campo e Interfaz.

FieldProperties.java **InterfaceProperties.java**	▪ Inserción de propiedades y metadatos del artefacto Entidad, Campo e Interfaz. ▪ Recuperación de propiedades y metadatos del artefacto Entidad, Campo e Interfaz.
EntityPropertiesKey.java **FieldPropertiesKey.java** **InterfacePropertiesKey.java**	Esta clase contiene el listado de claves de las propiedades de los artefactos Entidad, Campo e Interfaz. ▪ Recuperación de la clave de la propiedad de los artefactos Entidad, Campo e Interfaz.
FieldType.java **MetadaType.java**	Esta clase contiene funciones para insertar y recuperar el tipo del artefacto Campo o de los metadatos. ▪ Definición de tipo de artefacto Campo o de metadatos. ▪ Recuperación de tipo de artefacto Campo o de metadatos.
Metadata.java	Esta clase contiene la definición y las funciones para completar o recuperar la información del objeto Metadata. ▪ Definición de la información del objeto Metadata. ▪ Recuperación de la información del objeto Metadata.
GovernanceEntityImpl.java **GovernanceFieldImpl.java** **GovernanceInterfacesImpl.java**	Esta clase contiene funciones para interactuar con los artefactos Entidad, Campo e Interfaz del Catálogo de Referencia. ▪ Creación de artefacto Entidad, Campo e Interfaz en el Catálogo de Referencia. ▪ Búsqueda de artefacto Entidad, Campo e Interfaz del Catálogo de Referencia. ▪ Guardado de artefacto Entidad, Campo e Interfaz en el Catálogo de Referencia. ▪ Checkout de artefacto Entidad, Campo e Interfaz del Catálogo de Referencia. ▪ Deshacer checkout de artefacto Entidad, Campo e Interfaz del Catálogo de Referencia. ▪ Recuperación de la última versión de artefacto Entidad, Campo e Interfaz del Catálogo de Referencia.
GovernanceSchemaImpl.java	Esta clase contiene funciones para la carga y descarga de schema de una Entidad y de WSDL de una Interfaz. ▪ Carga del schema de un artefacto Entidad del Catálogo de Referencia. ▪ Descarga del schema de un artefacto Entidad del Catálogo de Referencia. ▪ Descarga del WSDL de un artefacto Interfaz del Catálogo de Referencia.
GovernanceServer.java	Esta clase contiene funciones que utilizan las clases (Governance) que interactúan con los artefactos Campo, Entidad e Interfaz del Catálogo de Referencia.

	▪ Creación de artefacto Entidad, Campo e Interfaz en el Catálogo de Referencia. ▪ Búsqueda de artefacto Entidad, Campo e Interfaz del Catálogo de Referencia. ▪ Guardado de artefacto Entidad, Campo e Interfaz en el Catálogo de Referencia. ▪ Checkout de artefacto Entidad, Campo e Interfaz del Catálogo de Referencia. ▪ Deshacer checkout de artefacto Entidad, Campo e Interfaz del Catálogo de Referencia. ▪ Recuperación de la última versión de artefacto Entidad, Campo e Interfaz del Catálogo de Referencia.
GovernanceAPI.jar	Paquete que agrupa funciones de integración con Registry, ya que las funciones nativas son extremadamente atómicas. ▪ Permite gestionar los elementos del Catálogo abstrayendo de los recursos que lo componen y de las operaciones realizadas. ▪ Tiene un fichero de propiedades WSO.properties que incluye el direccionamiento de la herramienta de registro.

Tabla 7. Módulos Principales de IDE Utils

4.3. Ficheros de Parametrización

IDE Utils contiene un conjunto de ficheros de propiedades que permiten su parametrización. En la siguiente tabla se enumeran y describen estos ficheros (ubicados en **/resources/properties/**):

Fichero	Descripción
BIAN.properties	Información sobre la topología BIAN implementada. La información que contiene es: - *Business Areas* - *Business SubAreas* - *Business Domains* - *Service Domains*
EntityMetadata.properties	Metadatos funcionales para un artefacto de tipo Entidad.
EntityType.properties	Tipos de Entidades (clasificación)
FieldType.properties	Definición de los tipos de Campos.
InterfaceMetadata.properties	Metadatos funcionales para un artefacto de tipo WSDL.

InterfaceType.properties	Definición de los tipos de Interfaces.
MetadataConstraints.properties	Definición de las incompatibilidades entre facets.
MetadataDescription.properties	Descripción de metadatos.
NomenclatureRules.properties	Reglas de nomenclatura para el modelado de servicios.
WsdlTemplatesId.properties	Versión de las plantillas que está utilizando el plug-in.

Tabla 8. Ficheros de Parametrización de IDE Utils

El modelado de Entidades e Intefaces Web se realiza a través de un sistema de **composición modular**. En la siguiente tabla se enumeran y describen las plantillas que facilitan el modelado (ubicados en **/resources/templates/**):

Fichero	Descripción
Entity.template	Plantilla de modelado de entidades
WSDLSchema.template	Plantilla modelo de la sección Schema para definir los mensajes de entrada/salida
WSDLPortType.template	Plantilla modelo de la sección PortType
WSDLBinding.template	Plantilla modelo de la sección Binding
WSDLMessage.template	Plantilla modelo de definición de mensajes de las operaciones
WrapperAdapter.wsdl **ChannelAdapter.wsdl** **BusinessService.wsdl**	Plantillas maestras de WSDL que embeben todas las anteriores.

Tabla 9. Plantillas de Modelado de WSDLs

4.4. Interfaz de Usuario

El punto de acceso es el menú de Eclipse "*Uc3m*" tal y como se muestra en la imagen siguiente*:*

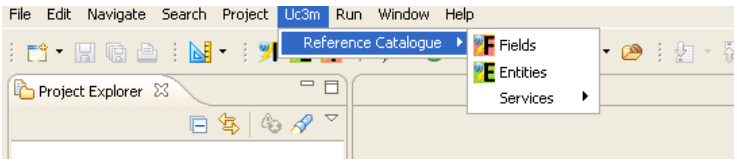


Ilustración 29. Menu de acceso

A continuación, se expondrán todas las ventanas que conforman la interfaz de usuario de la herramienta *IDE Utils*, agrupadas por elemento del Catálogo de Referencia al que operan:

1-CAMPOS:

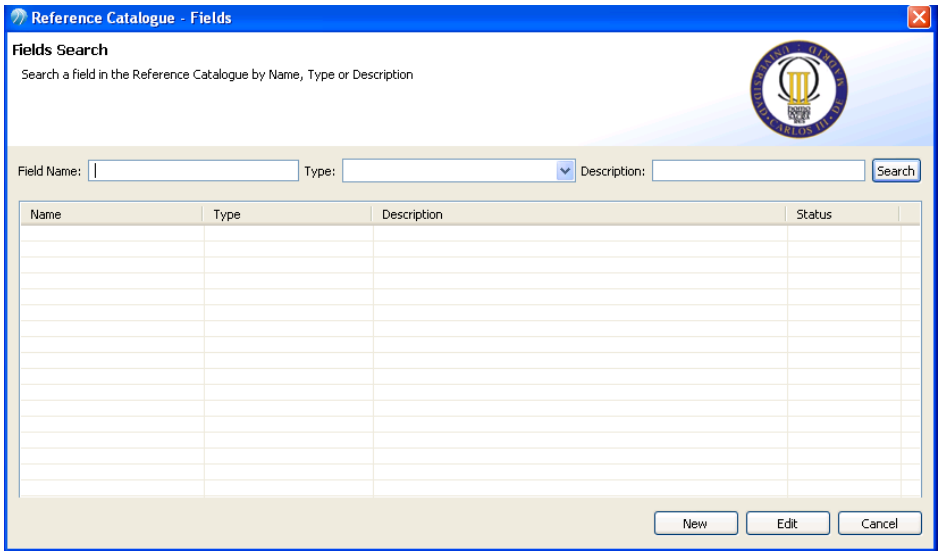


Ilustración 30. Ventana de búsqueda de campos

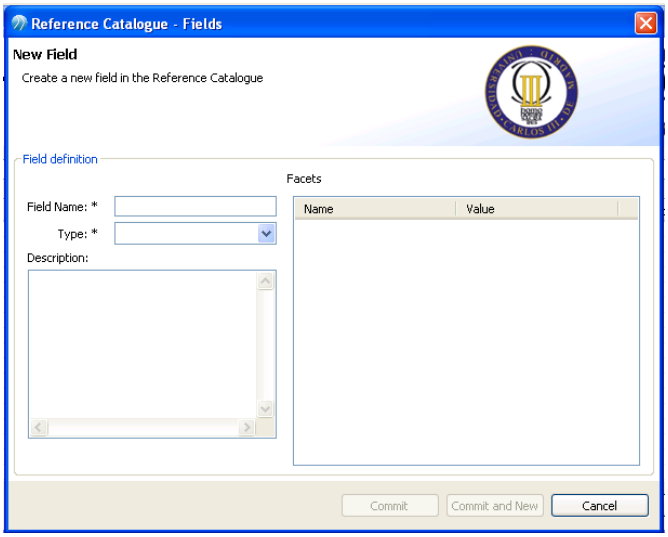


Ilustración 31. Ventana de creación/edición de campos

2-ENTIDADES:

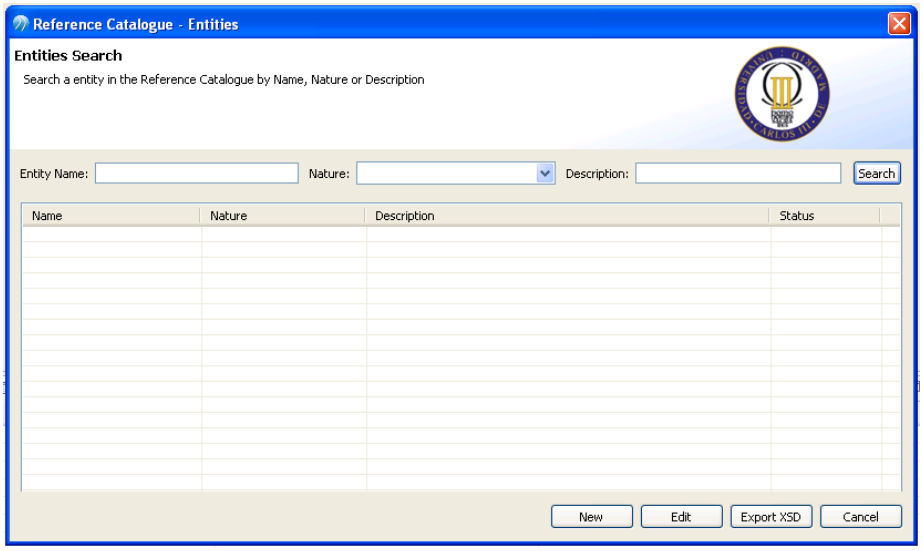


Ilustración 32.Ventana de búsqueda de entidades

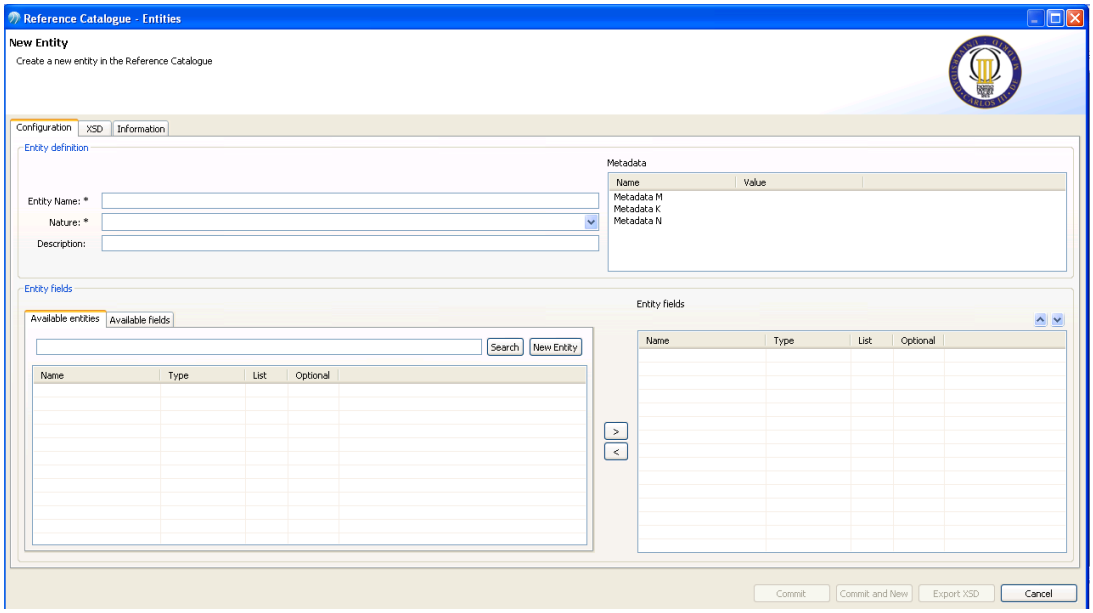


Ilustración 33. Ventana de creación/edición de entidades

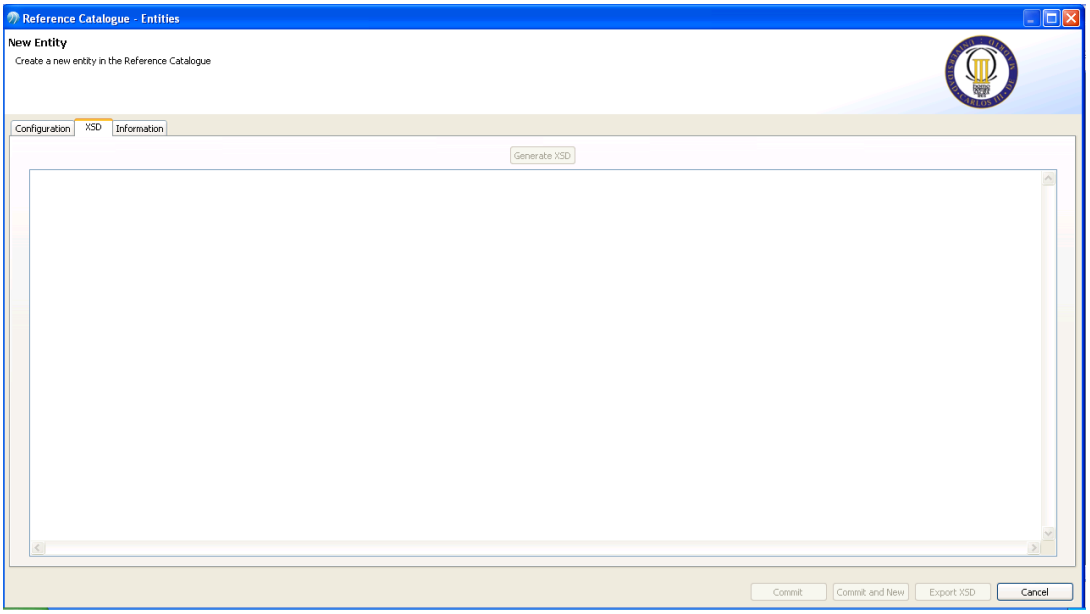


Ilustración 34. Ventana de visualización de XSD de entidades

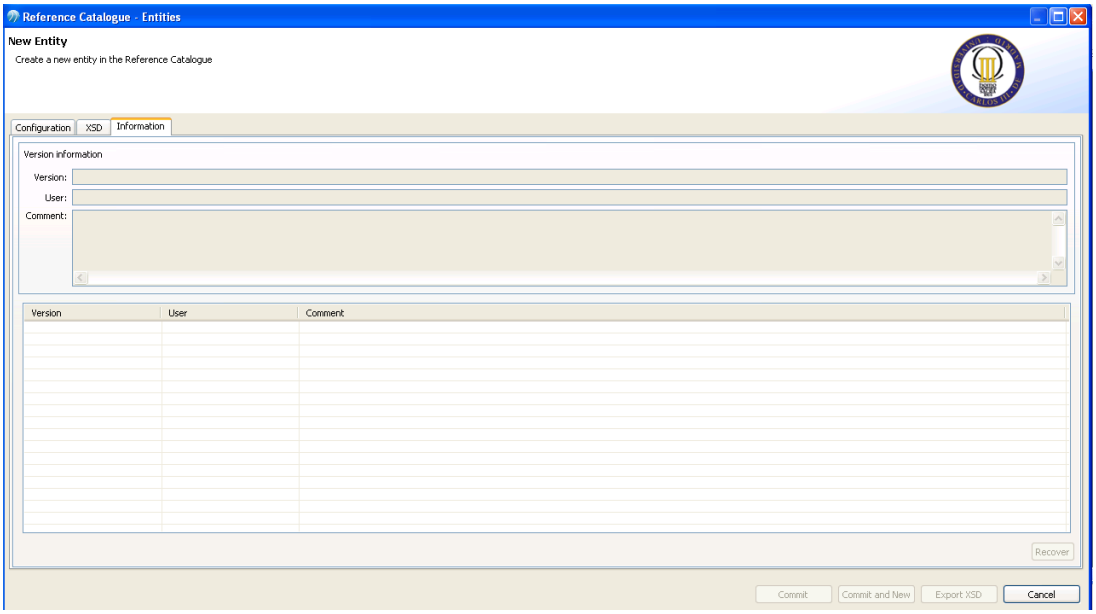


Ilustración 35. Ventana de metainformación de entidades

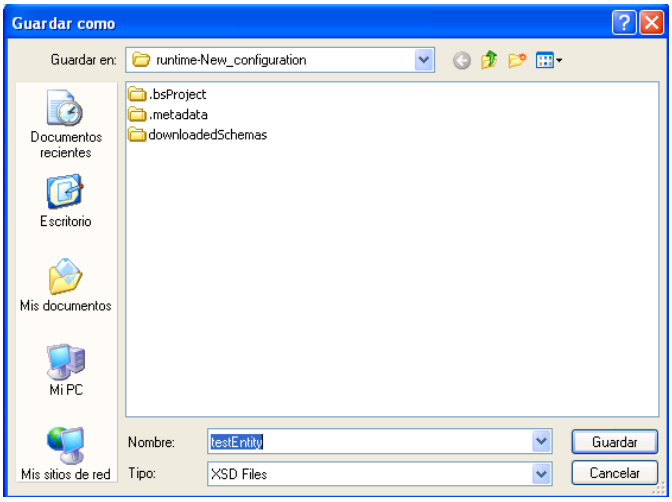


Ilustración 36. Ventana de exportación de XSD

3.-CONTRATOS DE SERVICIO (WSDL):

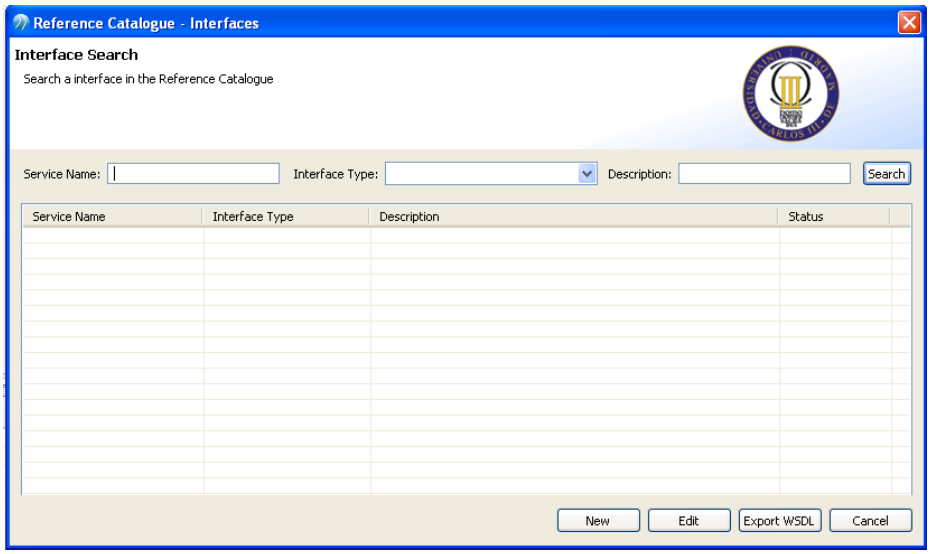


Ilustración 37. Ventana de búsqueda de servicios

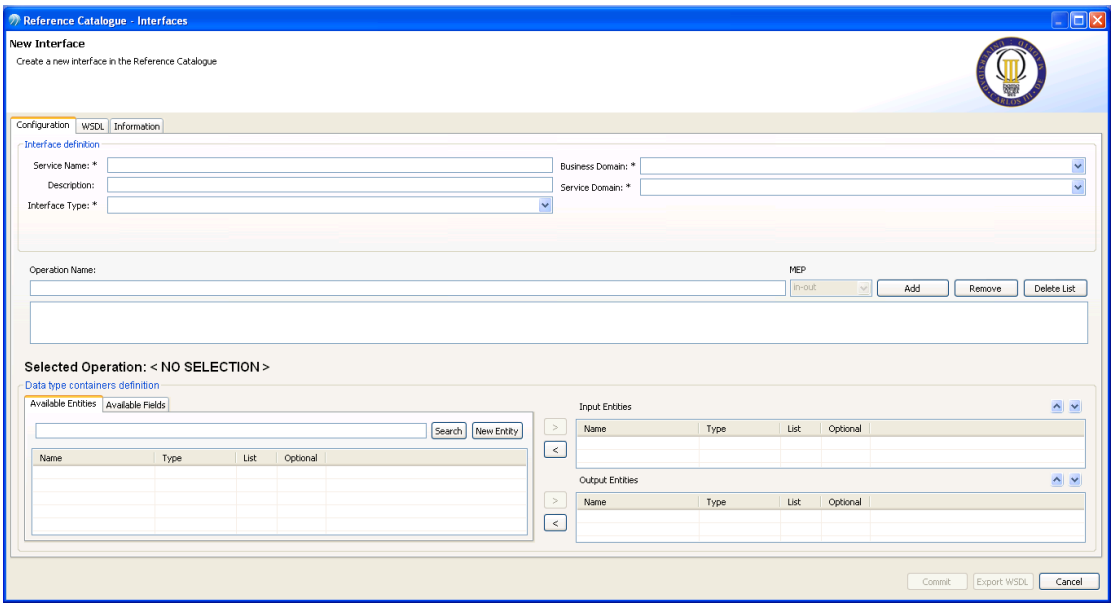


Ilustración 38. Ventana de creación/edición de interfaces

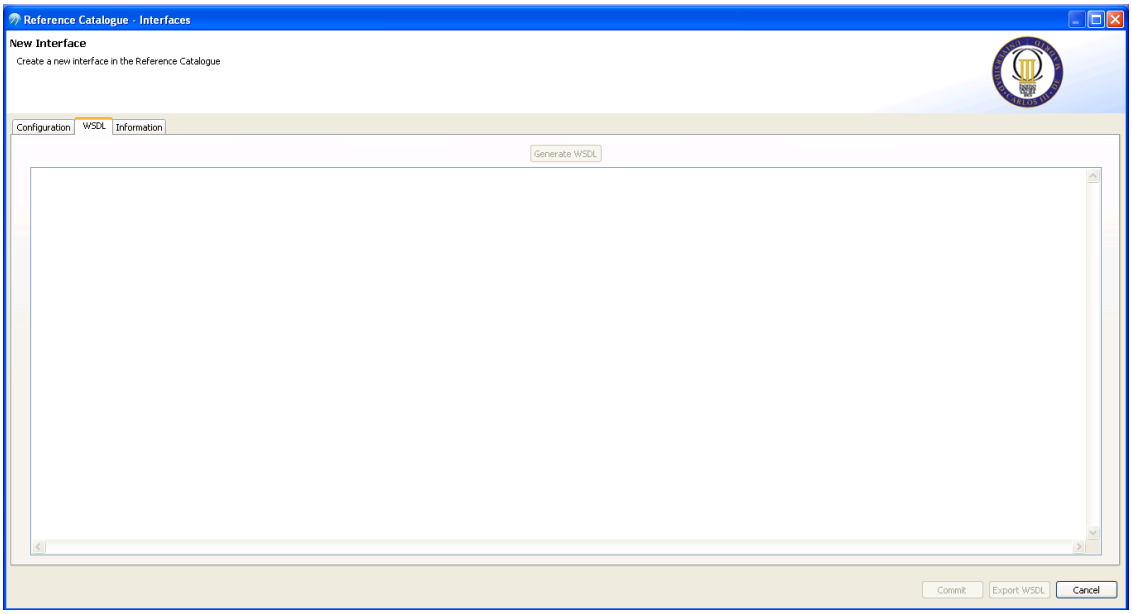


Ilustración 39. Ventana de visualización de WSDL

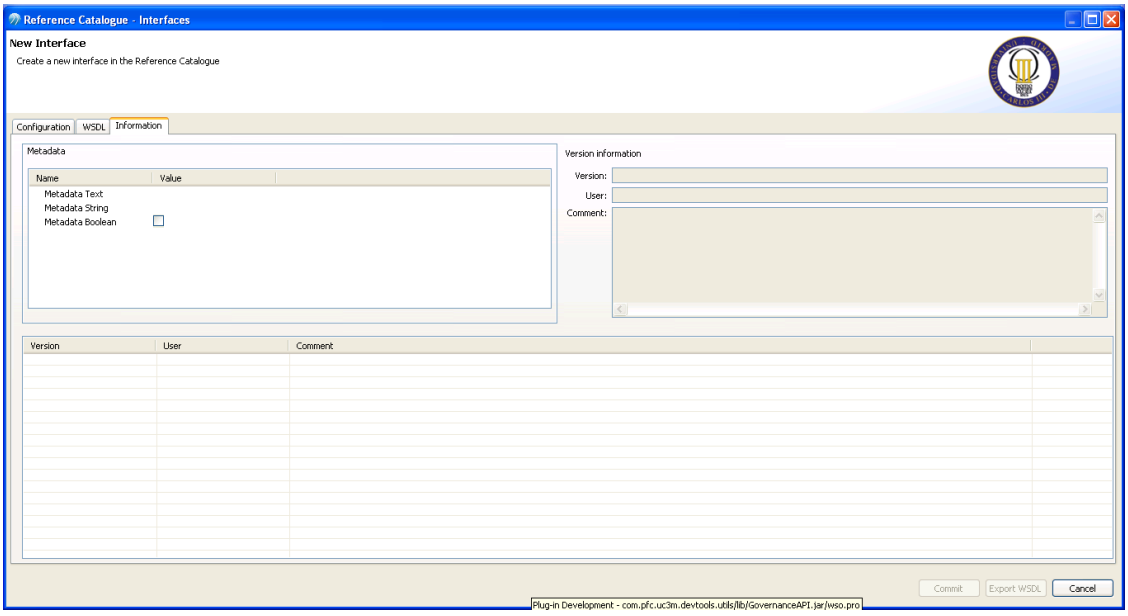


Ilustración 40. Ventana de metainformación de interfaces

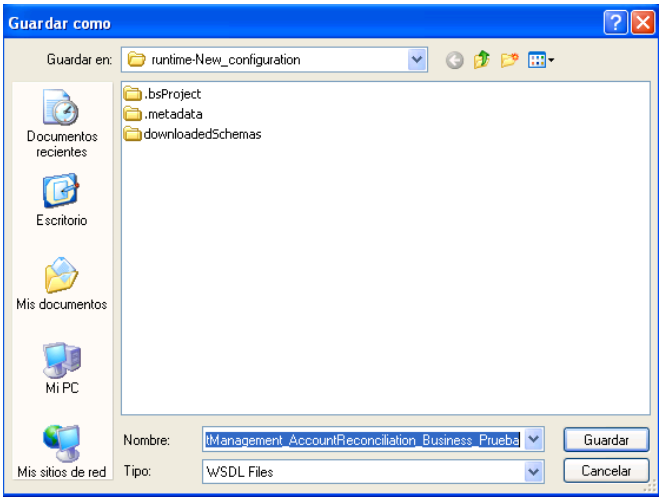


Ilustración 41. Ventana de exportación de WSDL

4.5. Manual de Referencia para el desarrollador

El *plug-in* es una utilidad desarrollada en lenguaje de programación *Java* y su desarrollo se realiza bajo un entorno *Eclipse*.

Este apartado está orientado a facilitar la extensión de la aplicación *IDE Utils* por parte de cualquier desarrollador: incluirá los detalles técnicos necesarios para la configuración del entorno de trabajo Eclipse, así como para la generación del "paquete jar" distribuible.

4.5.1. Configuración de Entorno de Trabajo

Para ampliar la funcionalidad de *IDE Utils*, el primer paso a seguir por parte del desarrollador es la importación del proyecto proporcionado en este PFC dentro del espacio de trabajo (*workspace*) del entorno *Eclipse*:

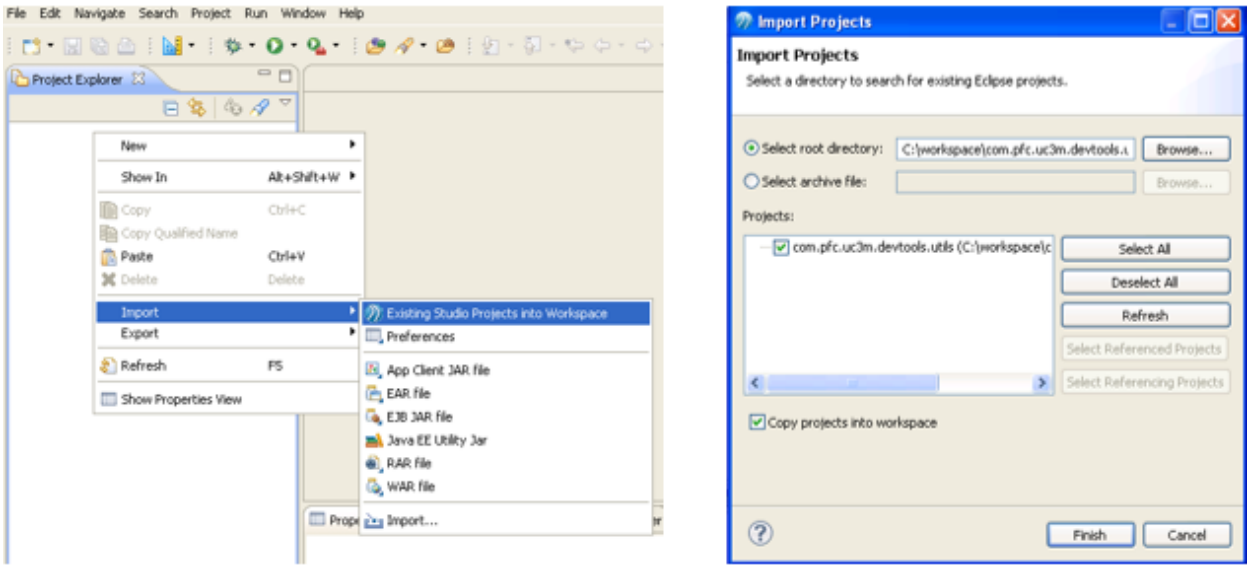


Ilustración 42. Importación de proyecto existente

Una vez importado el proyecto, existirán múltiples errores de compilación en el mismo debido a la no resolución de dependencias: para el desarrollo de *plug-ins Eclipse* es requerido que el parámetro *Target Platform* (dentro del menú de Eclipse: *Window > Preferences)* apunte exclusivamente a la *Running Platform*, la propia plataforma Eclipse.

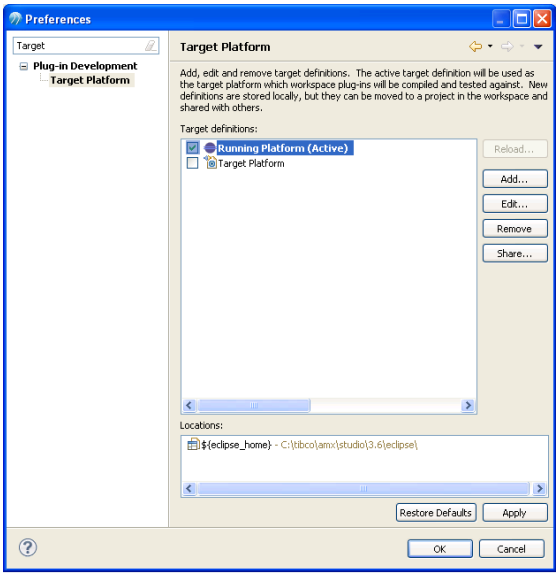


Ilustración 43. Cambio de Target Platform

4.5.2. Generación de paquete distribuible

La finalidad de toda utilidad *plug-in* es ser distribuida a todos potenciales usuarios de la misma. Para la consecución de tal objetivo la utilidad debe ser empaquetada como *plug-in Eclipse*.

Para realizar este empaquetado, se debe exportar el proyecto de desarrollo como "*Deployable plug-ins and fragments*" como se indica en las siguientes figuras:

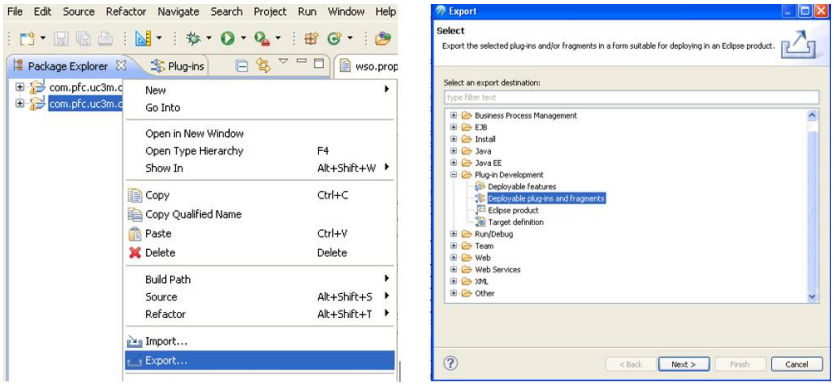


Ilustración 44. Exportación de proyecto como plug-in distribuible

Finalmente, el resultado de la operativa proporciona el fichero distribuible, que podrá ser incluido en los entornos de desarrollo *Eclipse* de los usuarios. En nuestro caso dicho fichero es:

com.pfc.uc3m.devtools.utils_1.0.0.jar

Capítulo 5

Conclusiones y Líneas Futuras

Todo proyecto debe finalizar con una reflexión sobre el trabajo realizado, exponiendo cuáles son las conclusiones extraídas de la realización del mismo e identificando nuevas iniciativas que permitan continuarlo.

"El instante es la continuidad del tiempo,
pues une el tiempo pasado con el tiempo futuro"

Aristóteles.

5.1. Conclusiones

Para tener una idea clara de los resultados obtenidos, haremos un breve recordatorio del objetivo perseguido por el proyecto: ***Normalización de los servicios de una organización y Modelado de servicios web***.

La realización del presente Proyecto Fin de Carrera ha promovido, en líneas generales, la consecución de los siguientes hitos:

- Proporcionar un Catálogo de Referencia único y centralizado, accesible por todas las áreas de desarrollo desde su entorno local, que albergue el glosario de datos de la organización, las entidades lógicas que compondrán los mensajes de los servicios y el catálogo de servicios disponibles.
- Facilitar al desarrollador el acceso al Catálogo de Referencia desde su entorno local de desarrollo Eclipse, permitiendo el mantenimiento del mismo y la consulta de las interfaces publicadas por todas las áreas.
- Proveer a la arquitectura de una utilidad de modelado de los contratos de servicios web en base a unos patrones previamente establecidos, y alimentada desde el Catálogo de Referencia de la organización.

Dicho esto, podemos concluir que se ha provisto a la arquitectura de una organización de la **capacidad de normalizar sus servicios**, mediante la utilización de un **lenguaje común** albergado en el Catalogo de Referencia (glosario de datos y entidades), y se ha proporcionado a analistas y desarrolladores una herramienta con doble funcionalidad: **mantenimiento del Catálogo de Referencia y modelado de servicios web.**

A continuación, concretaremos los beneficios de la inclusión, dentro del modelo de desarrollo, de un Catálogo de Referencia integrado con el entorno Eclipse:

- **Reducción de costes en los proyectos por minimización de errores:** la existencia de un glosario de datos y entidades centralizado evita los errores en la definición de los mensajes entre servicios, lo que se traduce en la reducción de costes en la ejecución de los proyectos.
- **Reducción de los costes en los proyectos por reutilización de componentes**: tener un registro de servicios permite que, en tiempo de diseño, los analistas puedan verificar, con criterios de negocio, si ya existe la funcionalidad que necesitan.
- **Homogeneización y abstracción de los servicios publicados por diferentes satélites:** de cara a integrar diferentes sistemas y tecnologías, el registro permite la implementación de una capa de mediación agnóstica, en la que los analistas no deben preocuparse de la lógica de los servicios invocados, solo de la forma de hacerlo, que está claramente definida en el registro.
- **Utilización de un lenguaje común entendible por todas las áreas:** aterriza el lenguaje utilizado por los analistas de negocio sobre un modelo técnico que es entendible por las factorías de desarrollo; adicionalmente, se unifica la ubicación de las entidades de negocio y las técnicas, lo que simplifica la explotación del Catálogo de Referencia.
- **Mayor agilidad ante cambios:** ante cambios críticos (como por ejemplo, en la longitud de un identificador como el DNI), facilita enormemente la identificación de los servicios afectados al estar registradas la relación elemento-entidad-servicio.

Por otro lado, la utilización de la herramienta *IDE Utils*, expuesta en el Capítulo 4, permite el modelado de los contratos de servicio en base a unos patrones previamente definidos por los responsables de la arquitectura y proporciona las siguientes ventajas:

- **Garantía de calidad en la definición de contratos de servicios web:** el equipo de arquitectura define los patrones de diseño de los servicios, así como la forma de componer los mismos (nomenclatura, formato de mensajes, operaciones, *port types, bindings…)* dentro de las organizaciones; este tutelado en el desarrollo, asegura el cumplimiento de los requerimientos de arquitectura y la calidad de los contratos definidos.
- **Reducción de costes de integración:** desde el entorno de desarrollo eclipse se pueden exportar todos los WSDL de la organización, por lo que mejoran los tiempos de integración; del mismo modo, se mejoran las capacidades de orquestación de servicios.

- **Mayor velocidad en mantenimientos correctivos**: el registro centralizado de servicios almacena todas las versiones de los contratos, permitiendo su recuperación y edición a través de la herramienta de modelado, lo que supone una mejora en tiempo a la hora de tener que realizar un desarrollo correctivo y, por tanto, también en coste.

De modo genérico, podemos concluir que cuando mayor sea una organización y más complejos y cambiantes sean sus sistemas, mayor será el beneficio de utilizar soluciones como la presentada en este Proyecto Fin de Carrera

5.2. Líneas Futuras

Existen una serie de iniciativas orientadas a complementar, mejorar o extender la funcionalidad del trabajo realizado en este Proyecto Fin de Carrera; a continuación, se presentan las principales líneas de trabajo a futuro:

- **Nuevos Artefactos** - La herramienta WSO2 Governance Registry tiene la capacidad de albergar la definición de cualquier tipo de elemento, pero actualmente soporta un numero limitado artefactos: "*Filed*", "*Entity*", "*Schema*" (XSD), *"WSDL/WADL"*… Una iniciativa de mejora consiste en ampliar la capacidad del WSO2 Governance Registry para que pueda almacenar otros tipos de artefactos muy utilizados en las grandes organizaciones: ***COPYs COBOL*** y ***XML nativo*****.**
- **Transformaciones XSD** - Una alternativa a la mejora propuesta en el punto anterior consistiría en añadir una funcionalidad de transformación a la herramienta *IDE Utils*, que permitiese obtener COPYs o XML nativo a partir de *Entities* previamente almacenadas en el registro en formato XSD.
- **Resolución de dependencias a través de importaciones en XSD y WSDL** – Por simplificación del modelo, actualmente se están incluyendo las dependencias, entre entidades o entidades y WSDL, a través de anidamiento recursivo; si bien esta solución es perfectamente válida en todas las casuísticas, es mucho más elegante implementar estas dependencias mediante *imports*, además de blindar frente a cambios las definiciones.
- **Ciclo de Vida** - La herramienta de registro permite el mantenimiento del ciclo de vida de los artefactos que alberga; actualmente no se está controlando el ciclo de vida de los artefactos, por lo que un analista no es capaz de identificar el ciclo de vida de los elementos. Si bien es cierto que este dato puede incluirse como *metadato* acompañando a los artefactos, es interesante poder controlarlo a través de la funcionalidad nativa de WSO2 Governance Registry (su frontal web permitiría una mejor experiencia de usuario).

- **Modelado de Roles** – Es la propia herramienta IDE Utils la encargada de registrar el usuario de sesión local automáticamente en WSO2 Governance Registry con la primera conexión; todos estos usuarios están siendo registrados con perfil de administrador. Sin embargo, en muchas organizaciones existe una separación de funcionalidades en cuanto a la definición de WSDLs, ya sea por áreas funcionales o simplemente por competencias. Por todo esto, la implementación de un modelo de roles (ya soportado por *Registry*) en IDE Utils facilitaría la adaptación a todo tipo de organizaciones.
- **Integración con Herramientas de Versionado y Despliegue** – La herramienta de registro puede incluir personalizaciones a través de *handler* java, dichas implementaciones pueden ser lanzadas en tiempo de registro de los elementos. La inclusión de Governance Regisry con el resto de herramientas de control del ciclo de vida (repositorios o gestores de despliegue) añade la robustez necesaria al modelo para el gobierno de las aplicaciones de una organización: garantía de calidad, alineamiento del ciclo de vida de todos los componentes, etc.

En la actualidad, existen productos de diferentes proveedores (*IBM Rational, TIBCO Software…*) que proporcionan capacidades de registro de servicios, pero ninguna de ellas es capaz de cubrir de forma nativa todas las necesidades de una gran organización; están enfocadas a la resolución de las necesidades de la propia *suite*. En cualquier, caso ninguna solución del mercado plantea la resolución de manera centralizada de la problemática expuesta en el presente Proyecto Fin de Carrera, por lo que se trata de un ámbito sin explorar y con múltiples huecos a completar.

Capítulo 6

Metodología y Valoración Económica

"No, no creo en la suerte,
pero sí en asignar valor a las cosas"

John Forbes Nash.

6.1 Metodología de Trabajo

Para la ejecución de este proyecto se ha planteado una metodología de trabajo con un enfoque ágil, que proporcione la flexibilidad requerida en un proyecto de estas características y permita una evaluación periódica de los resultados por parte del tutor.

Metodologías como *Scrum* están pensadas para trabajo colaborativo, en equipo; sin embargo, su filosofía es perfectamente aplicable a nuestro caso: 1) toma de decisiones en base a la experiencia, 2) aproximación a la solución a través de iteraciones (bloques temporales cortos y fijos), 3) desarrollo incremental para optimizar la previsibilidad y el control del riesgo, 4) demostraciones periódicas de los resultados.

El trabajo realizado ha sido ejecutado aplicando las siguientes consideraciones:

- **SPRINTs (iteraciones)**: Objetivos funcionales a alcanzar por periodos fijos de 2 semanas, implementando los cambios requeridos en la solución de forma incremental y revisando los avances periódicamente (control del riesgo).
- **ENTORNO**: El código para cada uno de los bloques se desarrolla sobre infraestructura proporcionada por el proyectante, quien facilita el soporte y la supervisión de las implementaciones.
- **COMUNICACIÓN**: Una comunicación ágil y fluida entre el alumno y el tutor permite aprovechar el conocimiento específico, facilitando la generación de una solución de alta calidad y ejemplar en cuanto a buenas prácticas.

Los objetivos establecidos para cada *sprint* son orientativos, estimados a priori, habiéndose realizado una **retrospectiva** al final de cada uno de los mismos para analizar el trabajo realizado y **replantear**, en caso necesario, los siguientes *sprints*. Es decir, en la práctica los objetivos concretos para cada iteración han sido fijados justo al comienzo de la misma.

Adicionalmente, se han realizado revisiones periódicas de la forma de trabajar para solventar déficits en el proceso de comunicación, prácticas ineficientes, etc., y contribuir de este modo a la mejora continua.

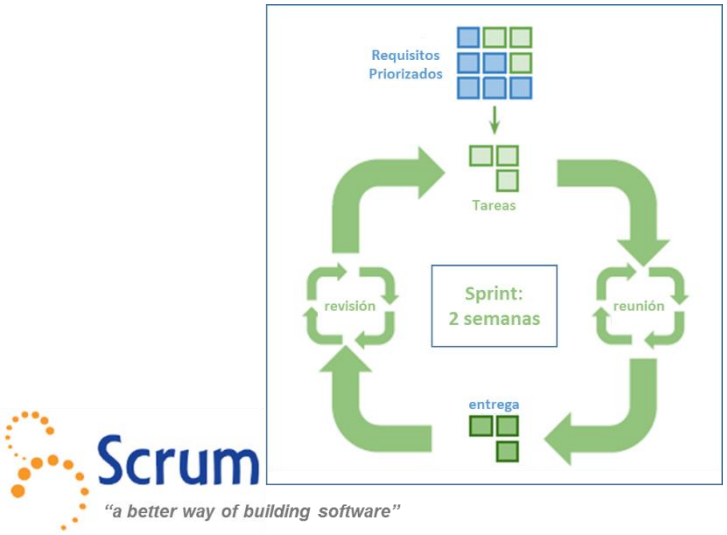


Ilustración 45.Metodología Scrum

6.2 Valoración Económica

6.2.1. Consideraciones Generales

La valoración económica de un proyecto viene determinada por el coste de los recursos utilizados para la ejecución del mismo más un margen de rentabilidad. Adicionalmente, el coste en cuestión se verá incrementado con la carga impositiva correspondiente: 21% IVA.

Se consideran dos tipos de partidas presupuestarias:

1) **Recursos Humanos**: Costes del esfuerzo de ejecución de las diferentes tareas de definición e implementación de las soluciones propuestas. La tarificación de estos costes variará en función del perfil y de las tareas realizadas, pudiéndose calcular un precio medio fijo (en inglés "fix price") para todas ellas.

2) **Recursos Materiales**: Costes relativos a gastos fijos como espacio, licencias, equipos, etc. Normalmente estos costes son considerados un gasto recurrente, (mensual o anual) y, en caso contrario, debería calcularse la amortización de costes de cada concepto, según índices establecidos, en función de la duración del proyecto.

6.2.2. Presupuesto de Proyecto

A continuación, se presenta la estimación del esfuerzo de ejecución del presente Proyecto Fin de Carrera en función de sus fases de ejecución:

Etapa	Descripción	Horas	Tarifa
Fase 1	Análisis de la problemática	160	1
Fase 2	Diseño y Valoración de la Solución	200	1
Fase 3	Planificación de la Ejecución	24	1
Fase 4	Construcción y Pruebas Unitarias	1008	2
Fase 5	Pruebas Integradas/Aceptación	432	2
Fase 6	Memoria de Proyecto	120	2

Tabla 10. Esfuerzo de ejecución

De la tabla anterior se desprende que el tiempo total dedicado por el proyectante ha sido de **1.944 horas**, a las que habría que sumar las dedicadas por el tutor; estas últimas no serán consideradas en esta valoración por estar remuneradas por otra vía.

El convenio colectivo de consultoría indica que "*la jornada ordinaria máxima de trabajo efectivo, en cómputo anual, será de 1.800.- horas anuales*", por lo que suponiendo una jornada laboral estándar de 40 horas semanales nuestro proyecto tendría una duración aproximada de **1 año y un mes**.

La siguiente tabla expone los gastos materiales, no atribuibles a esfuerzo humano (en adelante, *Otros Gastos*). Dichos gastos ascienden a **2.923 €** y aparecen desglosados en la siguiente tabla:

Concepto	Coste Recurrente	Coste Total (13 meses)
Puesto: - **Espacio/Mobiliario** - **Comunicaciones (internet)**	160€/mes	2080€
Licencias de Software: - **Oracle Java Development Kit 1.7.49** - **WSO2 Governance Registry 4.6.0** - **Oracle Database 11g R2 eXpress Edition** - **Eclipse Indigo (Release 3.7.2)**	0	0
Equipo Informático: - **PC (gama media con Win7 Pro)**	52€/mes	676€

Otros: - **Material de Oficina** - **Periféricos (grabadora, impresora…)** - **Soportes Digitales** - **Encuadernación**	167€ (pago único)	167€

Tabla 11. Coste de Recursos Materiales

La propuesta económica final se calcula a partir de los esfuerzos realizados en cada una de las fases de ejecución en base a las siguientes premisas:

- Se utilizarán tarifas de mercado en función de las fases, sin tener en cuenta jerarquiaspirámides al haber sido realizadas todas las tareas por el proyectante:
  - ✓ Tarifa 1 (Definición): **85€/h.** Para fases 1, 2 y 3
  - ✓ Tarifa 2 (Implementación): **47€/h**. Para fases 4, 5 y 6
- Los ciclos de prueba se han estimado como un 30% del esfuerzo total de implementación, excluida la fase de documentación.
- No se contemplan contingencias presupuestarias

Por tanto, el presupuesto empleado en la realización de este Proyecto Fin de Carrera es el que queda reflejado en la siguiente tabla:

Concepto	Esfuerzo (Horas)	Tarifa (€/hora)	Importe (€)
Definición	384	85	32.640
Implementación	1.560	47	73.320
Otros Gastos	—	—	2.923
TOTAL	**1.944**		**108.883**

Tabla 12. Coste de Proyecto

Resumen:

Los honorarios profesionales, correspondientes a los servicios que se describen en el presente documento, ascenderán a un total de ***ciento ocho mil ochocientos ochenta y tres euros*** (**108.883€),** quedando en vigor el precio de los servicios de la presente propuesta**.** Adicionalmente, se facturará los impuestos que gravan estos servicios: IVA 21%.

Bibliografía

[1]José Ñáñez Gómez (IT Architect, IBM Global Services), *Importancia del Gobierno SOA en la Organización*, IBM developerWorks®, 02/10/2014

[2] Paul C. Brown, *TIBCO® Architecture Fundamentals*, Boston, MA: Pearson Education, Inc. (Addison-Wesley), 2011

[3] Cuadrantes Mágicos de Gartner https://www.gartner.com

[4] XML, XDS y WSDL, World Wide Web Consortium (W3C) http://www.w3.org/standards/

[5] Guías BIAN "How-To" https://bian.org/servicelandscape/ (Publicado 12 Mayo-2015)

[6] Documentación de WSO2 Governance Registry 4.6.0 https://docs.wso2.com/display/Governance460/WSO2+Governance+Registry+Documentation

[7] Metodología ágil SCRUM http://proyectosagiles.org/

[8] Guía Oficial de SCRUM http://www.scrumguides.org/scrum-guide.html (Actualizado 2015)

[9]Convenio Colectivo del sector TIC http://www.ccoo-servicios.es/archivos/tic/200903_XVI_ConvenioColectivoTIC.pdf

Anexos

Anexo A.

Instalación WSO2 Governance Registry

El objetivo de este anexo es identificar las acciones a realizar para la instalación de la herramienta WSO2 Governance Registry y su base de datos asociada (H2 o Oracle XE) en un entorno centralizado y administrado que proporcione las garantías necesarias para desempeñar la función que le corresponde (Catálogo de Referencia de Servicios y Entidades).

A.1 Instalación de WSO2 Governance Registry

A.1.1 Prerrequisitos

Requerimientos de sistema:

- Sistema Operativo: Cualquier plataforma compatible con ***JDK 1.6.*/1.7.****
- Memoria: Mínimo 2GB de RAM y 512MB de *heap size*
- Disco: 1 GB excluido almacenamiento de logs y base de datos

Software de terceros:

- Oracle Java Development Kit 1.6.24 or posterior
- Navegador Web con JavaScript habilitado

Apertura de puertos:

5222 - 8000 - **9443** - **9763** – 9767 - 9999 - 10389 - 10500 – 11111

Los puertos 9443 y 9763 son los puertos necesarios para el producto WSO2 Governance Registry, el resto de puertos son para la integración con otros productos de WSO2.

A.1.2 Instalación WSO2 Governance Registry

El producto debe ser descargado desde la web http://wso2.com/products/governance-registry (es necesario registrarse) y para su instalación será descomprimido en una ruta exclusiva a este propósito, que en adelante denominaremos **[PRODUCT_HOME]**. El producto ya estará instalado y listo para ser configurado.

A.1.3 Instalación Base de Datos

Adicionalmente, debido al volumen de información que potencialmente será manejado a través de un Catálogo De Referencia, la instalación del registro se realizará con una base de datos externa al propio aplicativo: se ha elegido una base de datos Oracle 11g R2 por ser una de las más extendidas en el mundo empresarial.

Si de un entorno de pruebas se tratase, WSO2 Governance Registry podría funcionar con su base de datos embebida H2. En este caso sería necesaria la creación de esquemas ni la configuración de la conexión a la base de datos.

A.2 Base de Datos

A.2.1 Creación de Esquema de Base de Datos

Para la creación del esquema de base de datos es necesario el uso del usuario administrador de Oracle (por defecto, para un OracleXE: system/oraclexe).

Se deben editar los ficheros tnsnames.ora y listener.ora para definir las direcciones de conexión del nuevo esquema de base de datos y, posteriormente, reiniciar el Oracle.

Para crear el usuario de base de datos y su esquema asociado, se usa la herramienta "sqlplus" abierta como SYSDBA (`[ORACLE_HOME]/config/scripts/sqlplus.sh sysadm/password as SYSDBA`) donde se deben ejecutar los siguientes comandos:

```
create user [USUARIO] identified by [PASSWRD] account unlock;
grant connect to [USUARIO];
grant create session, create table, create sequence, create trigger to [USUARIO];
commit;
```

Los requerimientos de almacenamiento necesarios para el esquema de base de datos son:

- 3000 MB para datos
- 500 MB para índices

A.2.2 Configuración de la Base de Datos

En este caso la plataforma de base de datos es Oracle, para configurar la conexión a la base de datos hay que modificar el fichero "master-datasources.xml" que se encuentra en la ruta:

[PRODUCT_HOME]\repository\conf\datasources\

Habría que configurar los datasources WSO2_CARBON_DB y WSO2AM_DB, editando los siguientes atributos: usuario, password, url de conexión y driver, tal y como se indica en el siguiente bloque:

```
<datasource>
  <name>WSO2_CARBON_DB</name>
  <description>Datasource for registry and user manager</description>
  <jndiConfig>
    <name>jdbc/WSO2CarbonDB</name>
  </jndiConfig>
  <definition type="RDBMS">
    <configuration>
      <url>jdbc:oracle:thin:@SERVER_NAME:PORT/DB_NAME</url>
      <username>[USUARIO]</username>
      <password>[PASSWORD]</password>
      <driverClassName>oracle.jdbc.driver.OracleDriver</driverClassName>
      <maxActive>80</maxActive>
      <maxWait>60000</maxWait>
      <minIdle>5</minIdle>
      <testOnBorrow>true</testOnBorrow>
      <validationQuery>SELECT 1 FROM DUAL</validationQuery>
      <validationInterval>30000</validationInterval>
    </configuration>
  </definition>
</datasource>
```

En el caso de una configuración por defecto en Oracle Express Edition, los atributos quedarían:

```
<url>jdbc:oracle:thin:@ECSUWI90D:1521:XE</url>
<username>[USUARIO]</userName>
<password>[PASSWORD]</password>
<driverClassName>oracle.jdbc.driver.OracleDriver</driverClassName>
```

El nombre de los datasources podría modificarse aunque habría que modificar los siguientes ficheros de la misma carpeta para alinear la configuración:

- registry.xml
- user-mgt.xml
- identity.xml

A.2.3 Inicialización de Base de Datos

Si la variable JAVA_HOME no está definida como variable de sistema, hay que incluir su definición en el script de arranque del aplicativo:

Linux: `export JAVA_HOME=e:\tools\Java\jdk1.7.0_67\`

Windows: `SET JAVA_HOME=e:\tools\Java\jdk1.7.0_67\`

NOTA: Se recomienda versión de JDK superior a 1.6.0_24

Para evitar errores relacionados con la zona horaria, añadiremos la siguiente sentencia al fichero de ejecución de la herramienta, **[PRODUCT_HOME]**\bin\wso2server(.sh/.bat):

Linux: `export JAVA_OPTS="-Duser.timezone='+01:00'"`

Windows: `SET JAVA_OPTS="-Duser.timezone=Europe/Madrid"`

Es necesario copiar las librerías JDBC de Oracle utilizadas en la definición del "datasource" (en este caso: "**ojdbc6.jar**") en la ruta: **[PRODUCT_HOME]**\repository\components\lib\

Finalmente, para inicializar la base de datos se debe ejecutar el siguiente script en función del tipo de sistema operativo:

Windows: **[PRODUCT_HOME]**\bin\wso2server.bat **-Dsetup**

Linux: **[PRODUCT_HOME]**\bin\wso2server.sh **-Dsetup**

NOTA: Esto ejecutará el script SQL → **[PRODUCT_HOME]**\dbscripts\oracle.sql

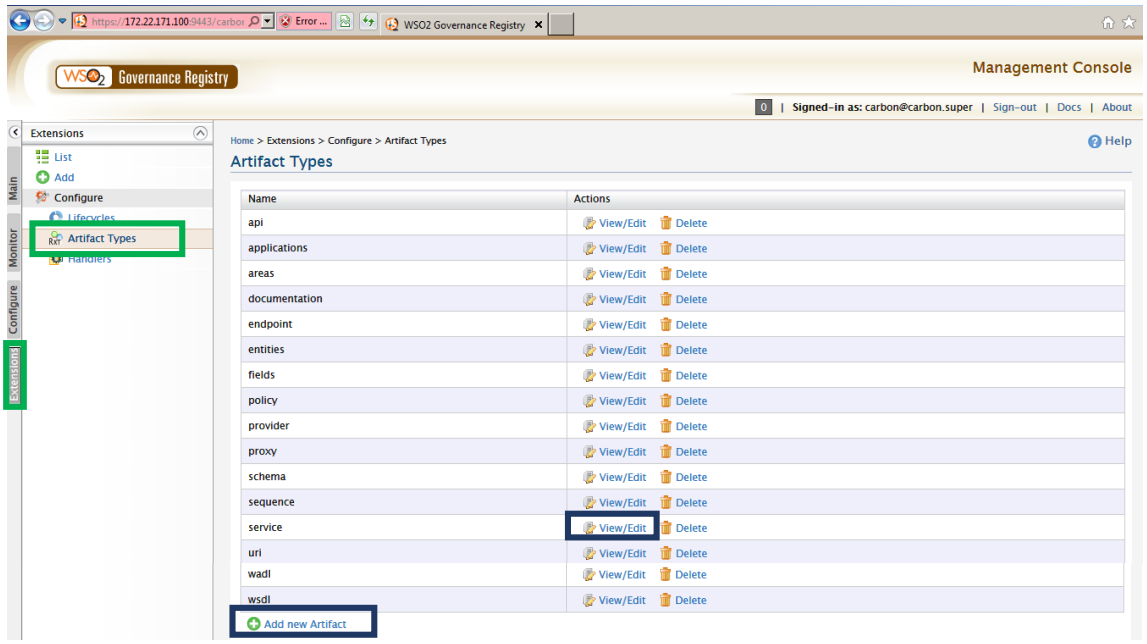
A.3 Carga de Elementos en WSO2 Governance Registry

A.3.1 Carga de Artefactos

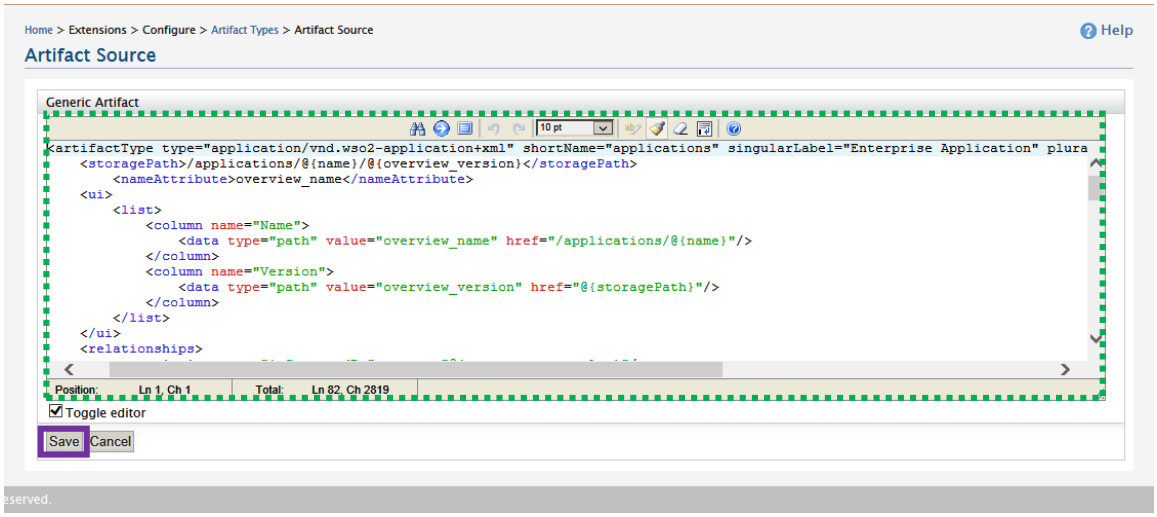
Para el Catálogo de Referencia se han definido una serie de "**artefactos a medida**" dentro del WSO2 Governance Registry; los artefactos definidos son los siguientes:

Field (nuevo)	**Entity (nuevo)**	**Service (modificado)**
fields.txt	entities.txt	service.txt

El mantenimiento de artefactos se realiza desde la consola de administración de WSO2 Governance Registry, a través de: ***Extensions > Artifact Types.***



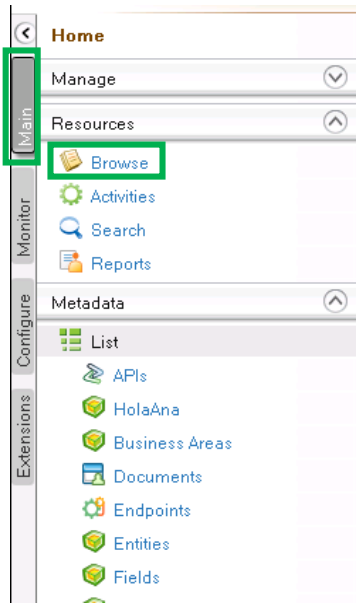
Los artefactos nuevos se añadiran a través de la opción ***Add new Artifact*** y para modificarlos a través de la opción ***View/Edit***; en cualquier caso, el procedimiento a seguir será la sustitución del xml de definición que aparece en el editor embebido en la consola por el proporcionado anteriormente en este apartado y salvar los cambios (***Save***)



A.3.2 Carga de Custom Queries

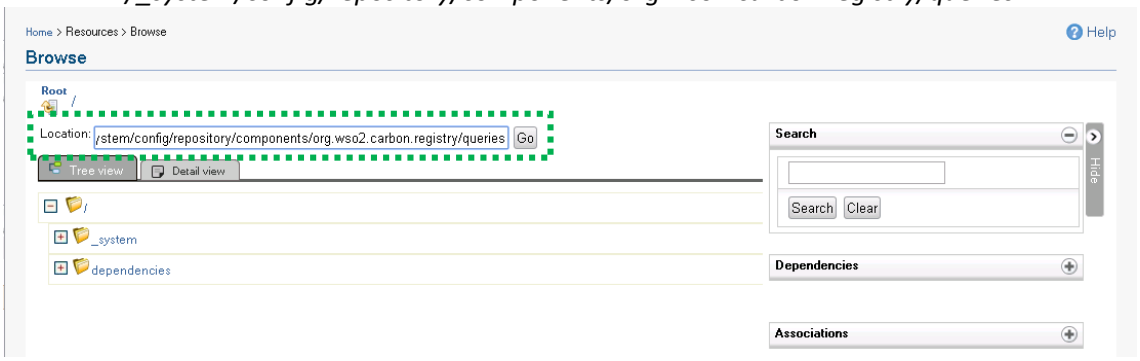
Para el Catálogo de Referencia se han definido una serie de "**consultas personalizadas**" dentro del WSO2 Governance Registry que se ejecutan para recuperar información referente a los artefactos del catálogo. Las *custom queries* a cargar son las empaquetadas en el siguiente fichero:

La carga de *custom queries* se realiza desde la consola de administración de WSO2 Governance Registry, a través de: ***Main*** **>** ***Browse*****.**

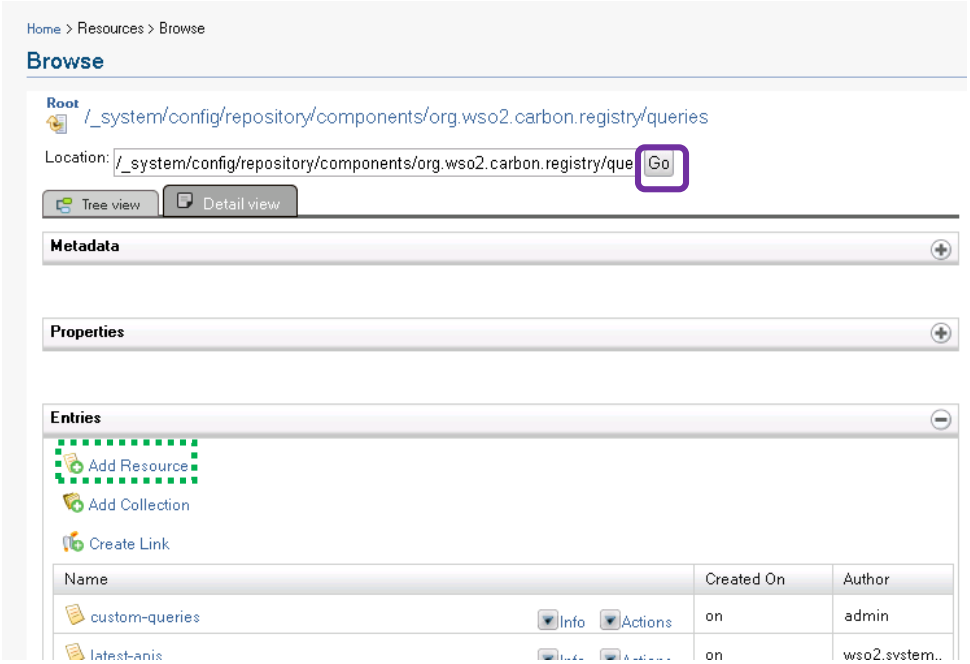


Al presionar ***Browse*****,** introducir en ***Location*** la siguiente ruta:

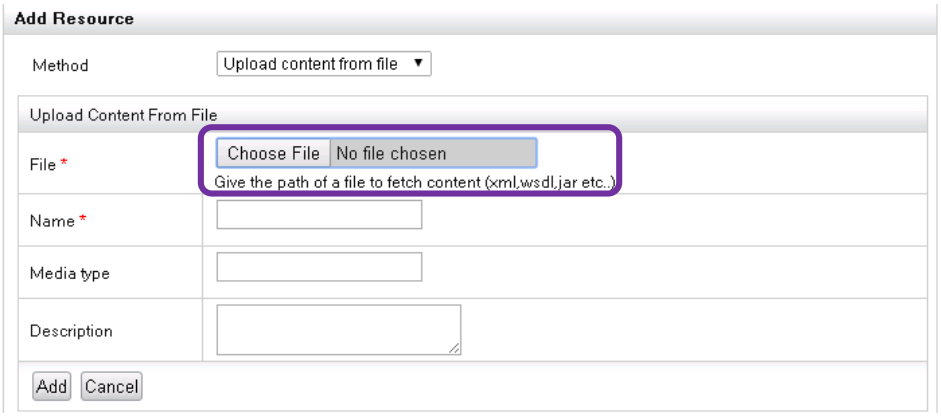
/_system/config/repository/components/org.wso2.carbon.registry/queries



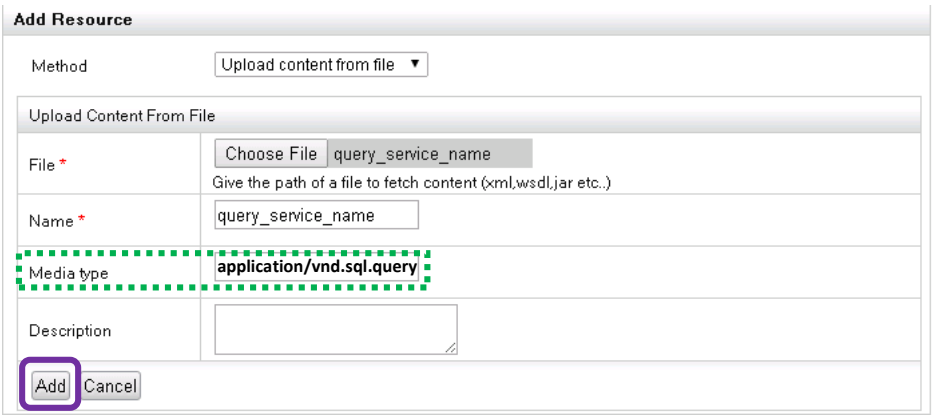
Presionar ***Go*** para ir a la ruta introducida y presionar ***Add Resource***.



Presionar ***Choose File***, navegar por el sistema de ficheros y seleccionar una a una las *custom queries* empaquetadas en el fichero anterior.



Cambiar el ***media type*** a ***application/vnd.sql.query*** y, finalmente, presionar ***Add*** para cargar la *custom query*.



Anexo B.

Guía de Uso de Consola Governance Registry

El objetivo de este documento es describir el uso de la consola de WSO2 Governance Registry para explotar la información del Catálogo de Referencia y, en particular, la relativa a las *customizaciones* realizadas en este proyecto. En el documento se describen los procedimientos para:

- Navegación cruzada entre los elementos del Catálogo
- Gestión de elementos del Catálogo:
  - ✓ Asociación manual de elementos
  - ✓ Bloqueo y desbloqueo de elementos
  - ✓ Borrado de elementos
- Publicación de recursos WSDL en el Catálogo

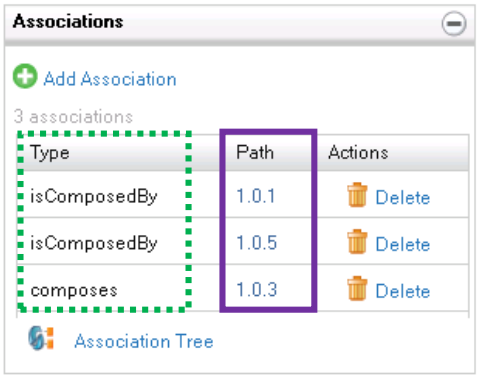
B.1 Navegación Cruzada de Elementos

Los elementos de WSO2 Governance Registry están asociados entre sí, por ejemplo, un WSDL puede estar compuesto por ninguna, una o varias Entidades que a su vez estén compuestas por otras Entidades o Campos. Gracias a esto, si se está consultando la información de un elemento desde la consola WSO2 Governance Registry es posible navegar por los elementos que lo componen o por los que él compone.

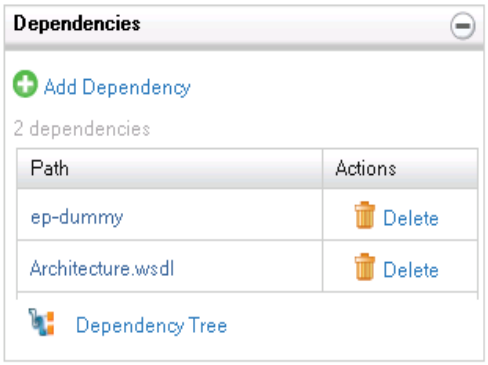
Las asociaciones entre elementos pueden visualizarse a través de la tabla **Associations** presente en la visualización de cada uno de los mismos. Se han definido dos tipos de asociaciones entre elementos:

- ***isComposedBy***: El elemento con esta asociación compone el elemento consultado.
- ***composes***: El elemento con esta asociación está compuesto por el elemento consultado.

Los enlaces de la columna ***Path*** permiten al usuario navegar a los elementos que componen o de los que forma parte el elemento consultado.



En la tabla ***Dependencies***, de cada elemento, el usuario puede consultar sus *imports* o *endpoints* además de poder navegar hasta ellos de la misma forma que lo hace para los elementos asociados.



B.2 Gestión de Elementos del Catálogo

Los elementos del Catálogo de Referencia se pueden gestionar desde la consola de WSO2 Governance Registry. Las funciones de gestión de elementos que explicaremos con más detalle son:

- Asociación manual de elementos
- Bloqueo y desbloqueo de elementos
- Borrado de elementos

B.2.1 Asociación Manual de Elementos

Desde la consola de WSO2 Governance Registry es posible asociar los elementos del Catálogo de Referencia manualmente, por ejemplo, asociar una Entidad con los Campos/Entidades que la componen.

Las asociaciones de los elementos WSDL y Entidad están reflejadas tanto en el artefacto de Registry como en la serialización DTCode del elemento.

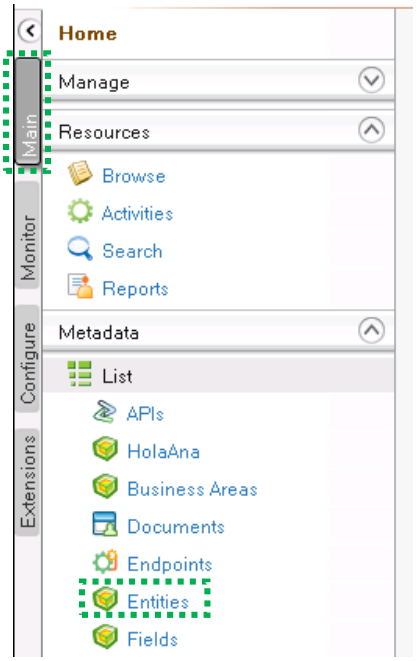
NOTA: Tener en cuenta que las asociaciones de Entidades y WSDLs con otros elementos se hacen a nivel de versión del elemento, por lo tanto, las asociaciones entre diferentes versiones pueden no mantenerse.

B.2.1.1 Asociación Entidad con Campo

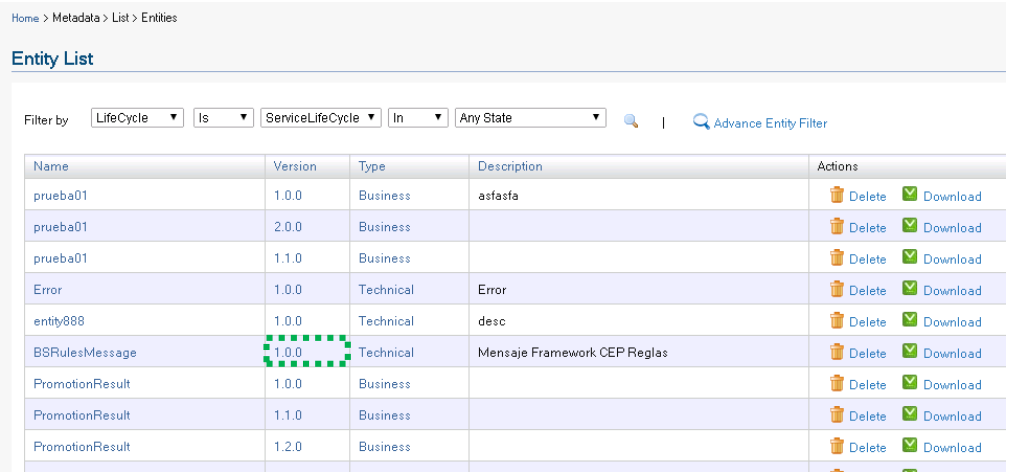
Una Entidad puede estar compuesta por uno o más Campos y por una o más Entidades. WSO2 Governance Registry permite crear asociaciones para reflejar las relaciones de composición entre los elementos del Catálogo de Referencia.

A continuación se detallan los pasos para asociar una Entidad con un Campo:

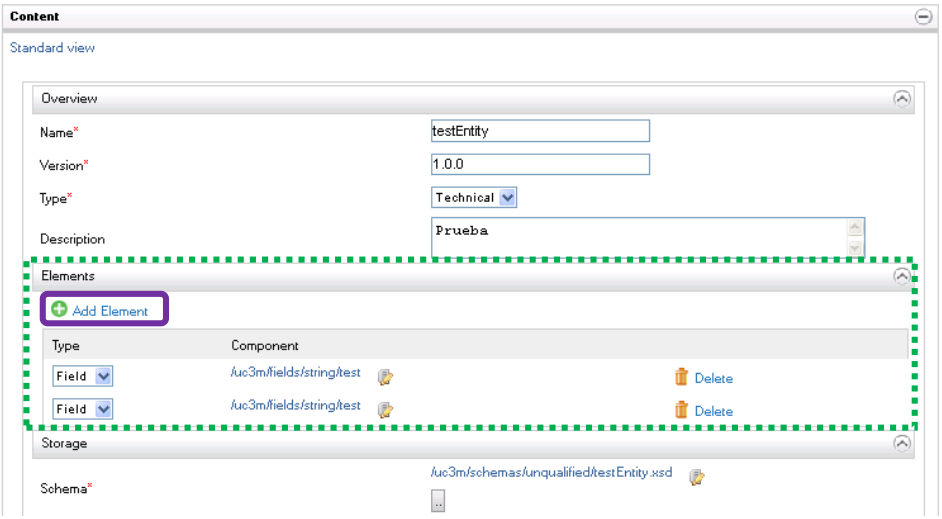
1. Presionar sobre ***Main*** **>** ***Entities***.



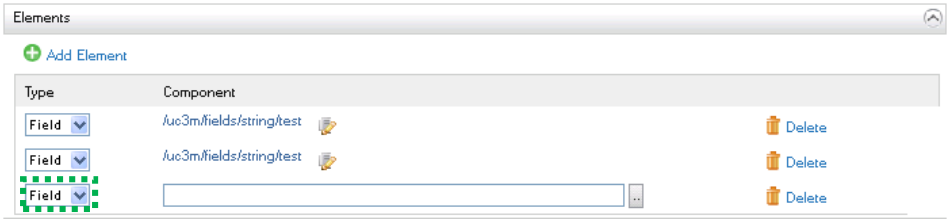
2. Buscar la Entidad a la que se le añadirá la asociación. Presionar sobre la ***versión*** de la Entidad a la que se quiere añadir la asociación.



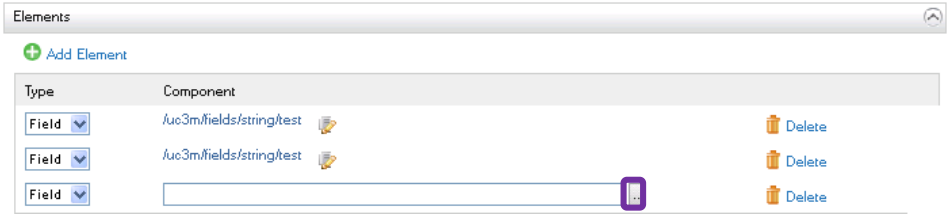
3. En la tabla ***Content > Elements*** presionar ***Add Element***.



4. Seleccionar ***Field*** o ***Entity*** en función del elemento que se quiera asociar a la Entidad.



5. Presionar el botón "***..***"

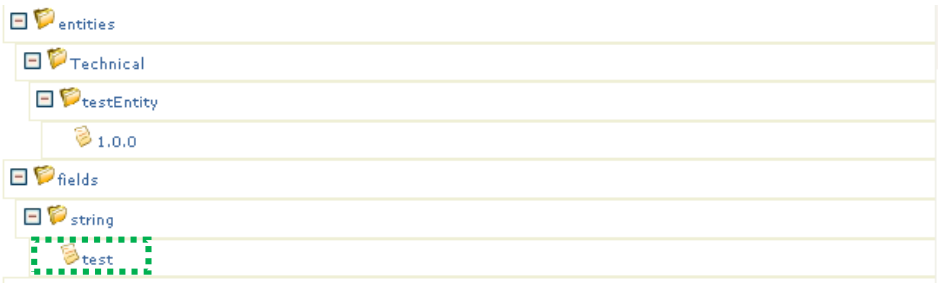


6. Navegar por el sistema de ficheros de WSO2 Gorvernance Registry. Las rutas de los Campos y de las Entidades en Registry son:

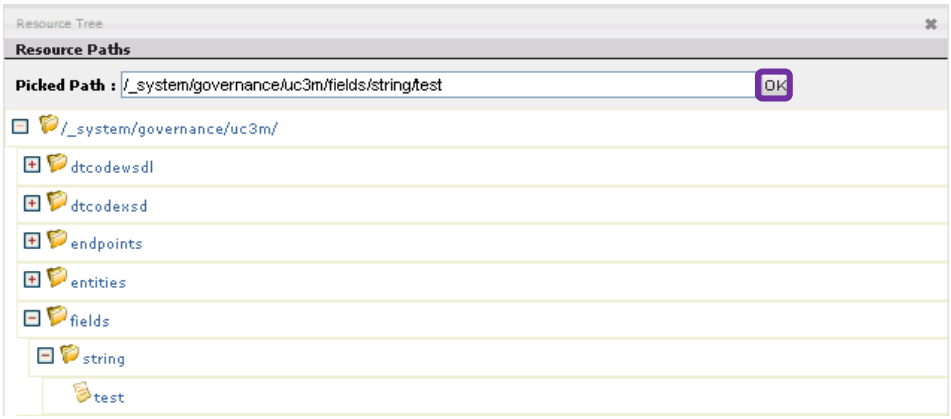
Elemento	Ruta
Entidad	*/_system/governance/uc3m/entities*
Campo	*/_system/governance/uc3m/fields*

Tanto los Campos como las Entidades del Catálogo de Referencia están organizados en subcarpetas según su tipo.

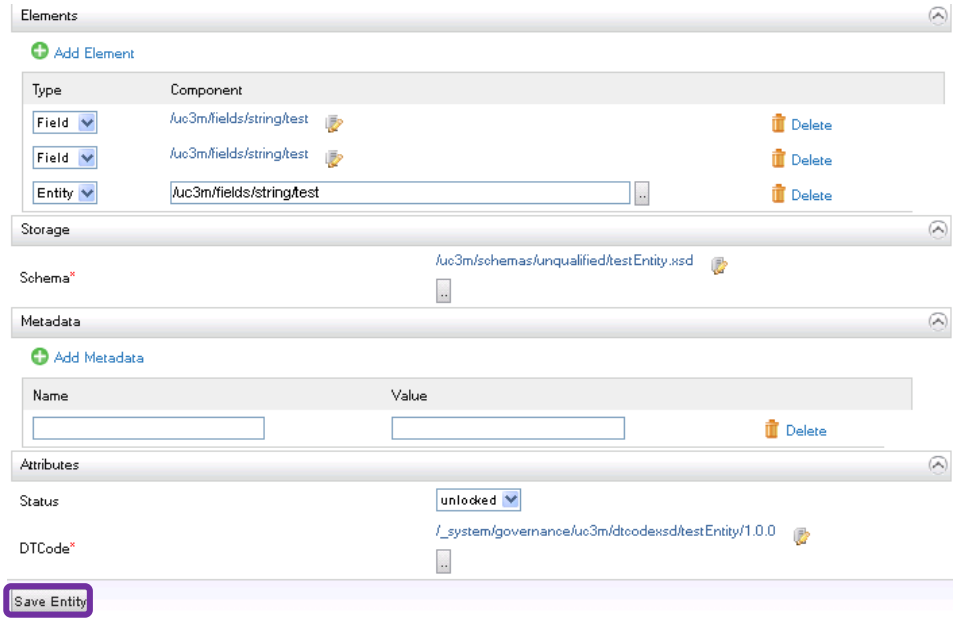
7. Una vez encontrado el ***campo***, en el sistema de ficheros de WSO2 Governance Registry, presionar sobre él.



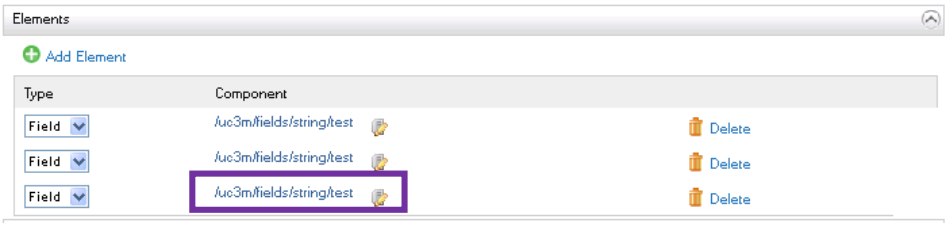
8. Presionar ***OK***.



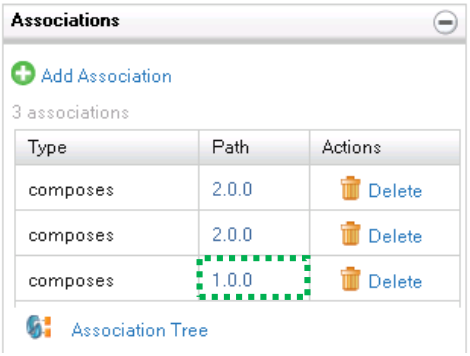
9. Presionar ***Save Entity***.



10. Para verificar que se ha creado la asociación, presionar el ***enlace*** del Campo añadido.



11. En la página del Campo, revisar las asociaciones de la tabla ***Associations*** para ver que aparece la ***entidad*** que le hemos asociado.

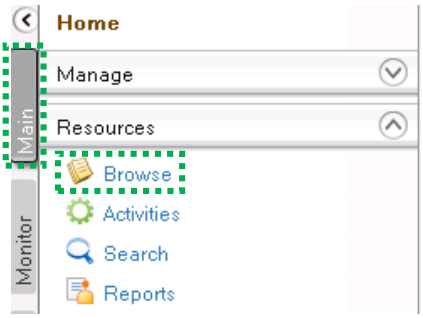


Modificación de DTCodeXSD

Las asociaciones de los elementos de WSO2 Governance Registry están reflejadas tanto en los artefactos del Catálogo como en el fichero DTCode, por lo tanto, también se debe modificar el DTCode para que éste incluya la asociación de los elementos.

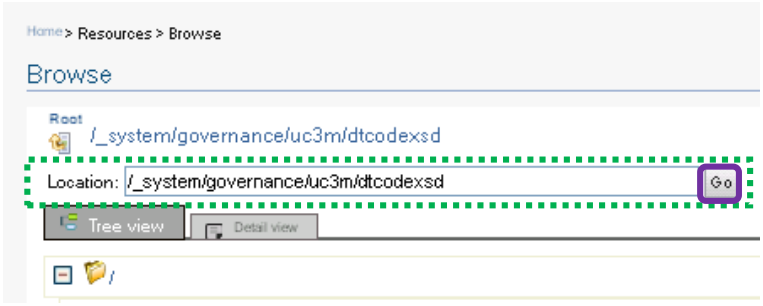
A continuación se detallan los pasos para añadir, en el DTCode, la asociación de un Field con una Entity.

1. Presionar sobre ***Main*** **>** ***Browse***.

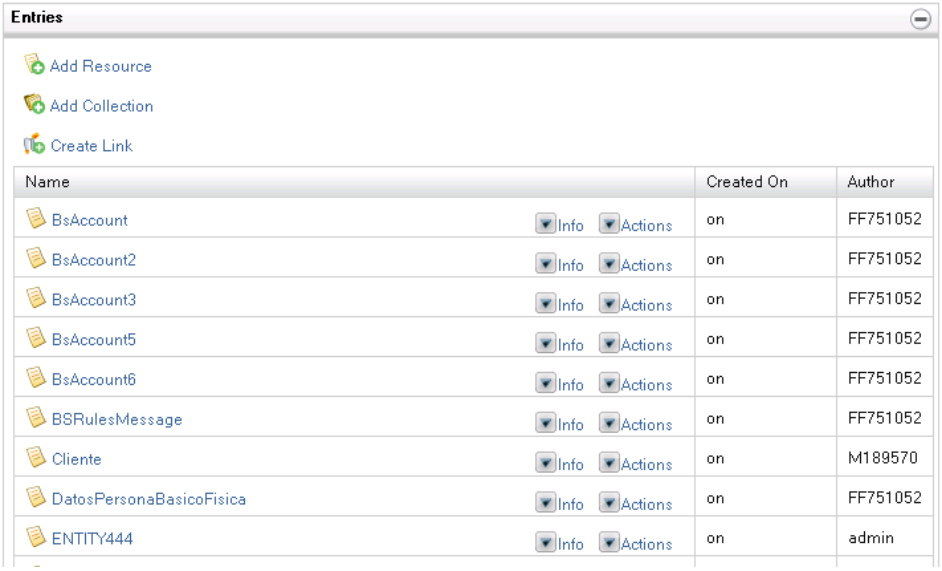


2. En ***Location*** pegar la siguiente ruta y presionar ***Go***:

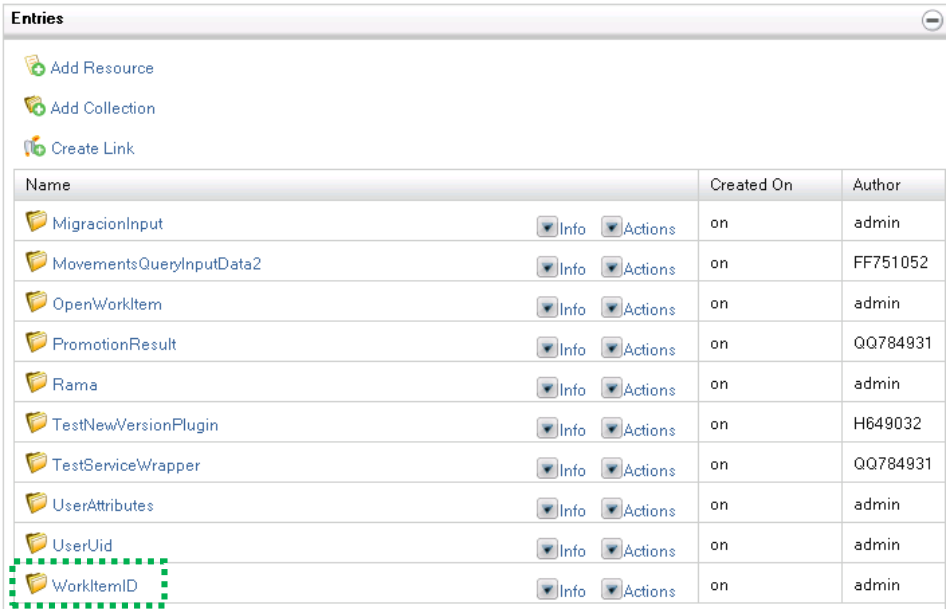
/_system/governance/uc3m/dtcodexsd



3. En la tabla ***Entries***, navegar por los ficheros hasta encontrar el directorio que contiene las versiones del DTCode correspondiente al artefacto Entity.



4. Una vez encontrado el directorio de DTCodes correspondiente a la entidad, presionar sobre él.



5. Desplegar ***Actions*** y presionar ***Download*** sobre la versión del DTCode que se desea modificar.

6. Con el fichero DTCode descargado, abrirlo con un editor de texto y añadir el código JSON del Field que se asociará.

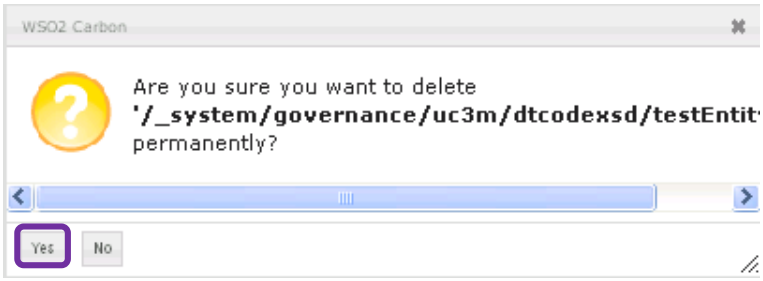
   Ejemplo de serialización de un *Field*:

```
{
 "@class":"field",
   "fieldType":{
    "code":"string",
    "metadata":[]
   },
   "name":"workItemID",
   "type":"string",
   "description":"ID del work item",
   "list":false,
   "optional":false,
   "version":null,
   "attributes_path":"/_system/governance/uc3m/fields/stri
ng/workItemID"
},
```

NOTA: El código anterior corresponde a la definición del Field y se debe añadir bajo el array JSON **children** del DTCode de la Entity.

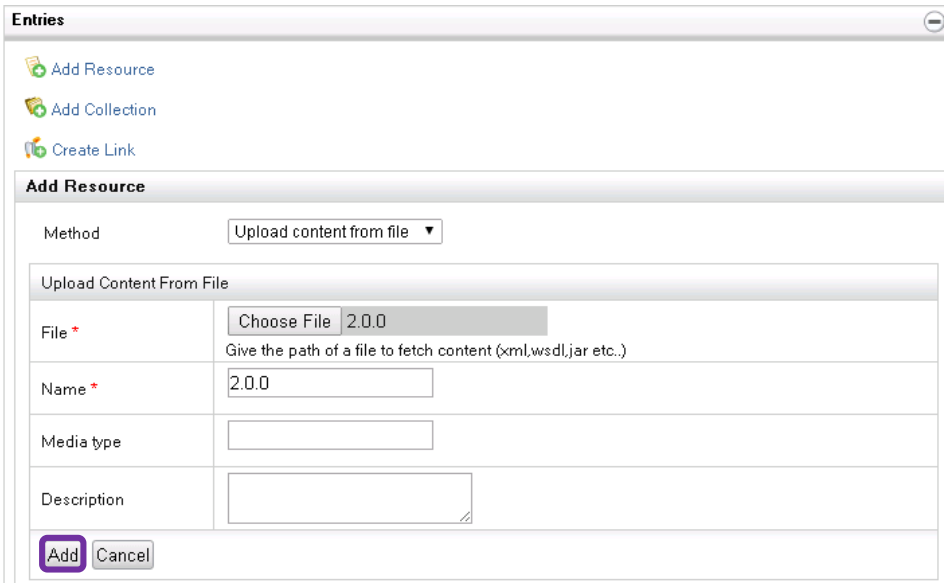
7. Guardar el fichero y borrarlo de WSO2 Governance Registry, presionando ***Delete***.

8. Presionar ***Yes*** en el diálogo de confirmación de borrado.

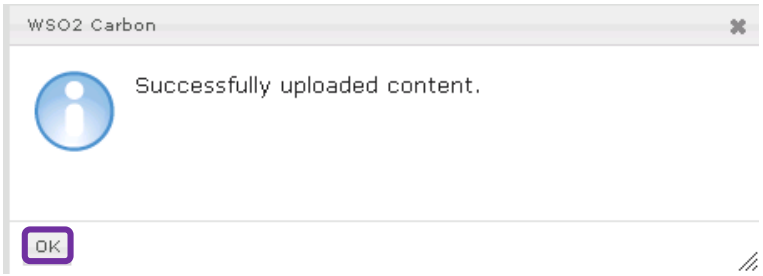


9. Presionar ***Add Resource*** para cargar en Registry el DTCode modificado localmente.

10. Navegar por el sistema de ficheros, seleccionar el DTCode donde se ha añadido la asociación manualmente y presionar ***Add***.



11. Finalmente, aparecerá una ventana confirmando que se ha cargado el fichero DTCode. Pulsar el boton ***OK*** para cerrar la ventana.

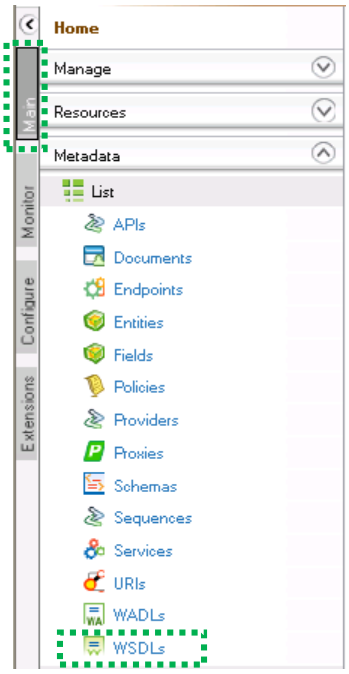


B.2.1.2 Asociación WSDL con Entidad/Campo

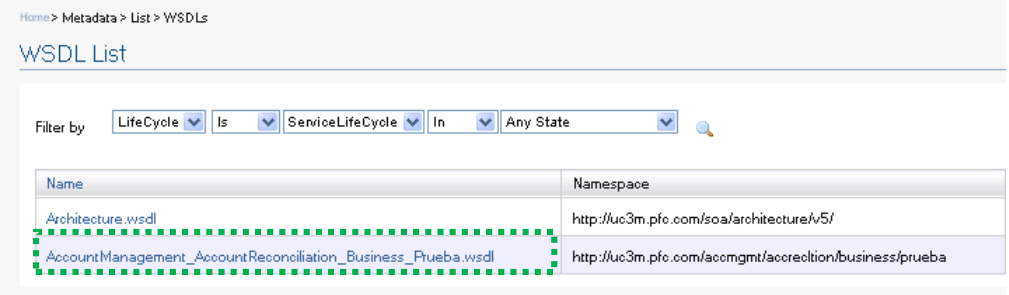
Un WSDL puede estar compuesto por ninguno, uno o varios Campos y por ninguna, una o varias Entidades. WSO2 Governance Registry permite crear asociaciones para reflejar las relaciones de composición entre los elementos del Catálogo de Referencia.

A continuación se detallan los pasos para asociar un WSDL con una Entidad/Campo:

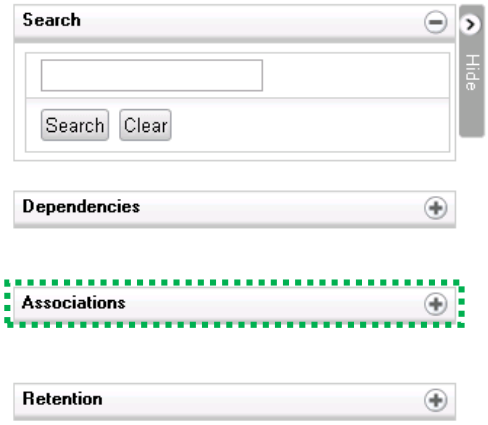
1. Presionar sobre ***Main*** **>** ***WSDLs***.



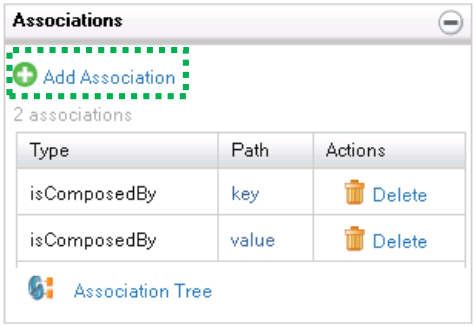
2. Buscar el WSDL al que se le añadirá la asociación y presionar sobre él.



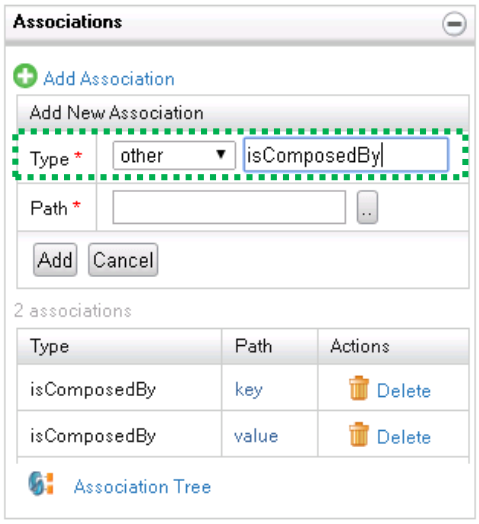
3. Desplegar la tabla ***Associations***.



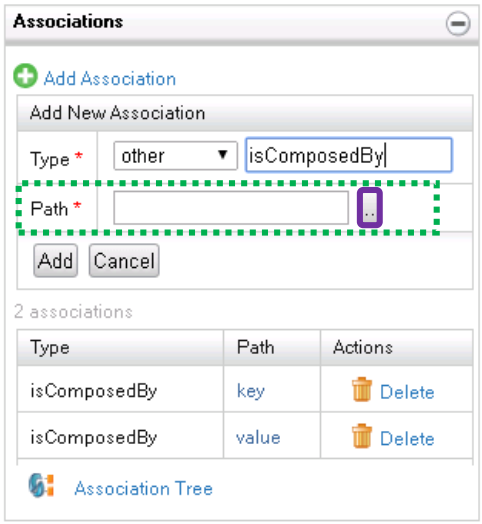
4. Presionar ***Add Association***.



5. En ***Type*** seleccionar ***other*** y escribir ***isComposedBy*** para que un Campo o Entidad forme parte de la Entidad que se está modificando.



6. En ***Path*** presionar el botón "***..***"

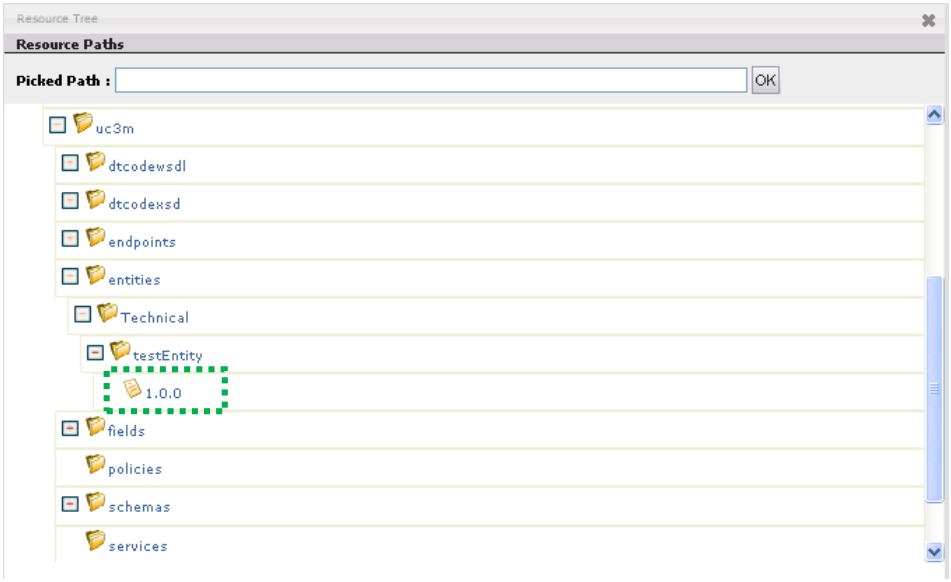


7. Navegar por el sistema de ficheros de WSO2 Gorvernance Registry. Las rutas de los Campos y de las Entidades en Registry son:

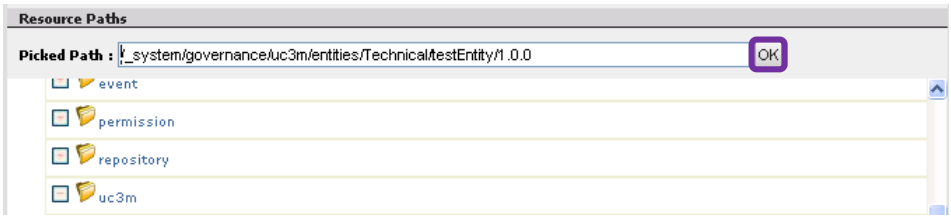
Elemento	Ruta
Entidad	/_system/governance/uc3m/entities
Campo	/_system/governance/uc3m/fields

Tanto los Campos como las Entidades del Catálogo de Referencia están organizados en subcarpetas según su tipo.

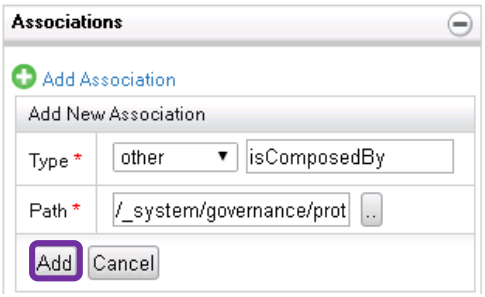
8. Una vez encontrada la Entidad o Campo en el sistema de ficheros de WSO2 Governance Registry presionar sobre la ***versión*** que se quiere asociar, en el caso que sea una Entidad, de lo contrario presionar directamente sobre el Campo.



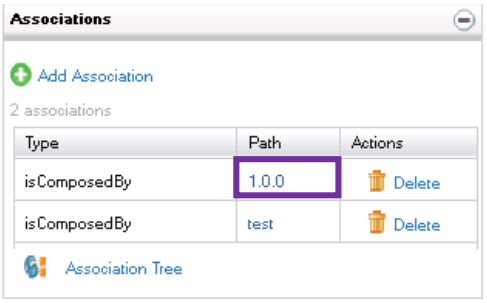
9. Presionar ***OK***.



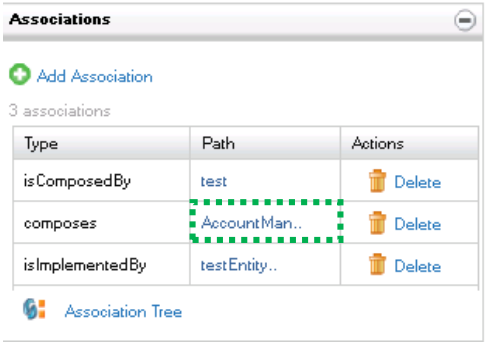
10. Presionar ***Add***.



11. Para verificar que se ha creado la asociación, presionar el ***enlace*** del Campo/Entidad añadido en la tabla ***Associations***.



12. En la página del Campo/Entidad, revisar las asociaciones de la tabla ***Associations*** para ver que aparece el WSDL asociado.

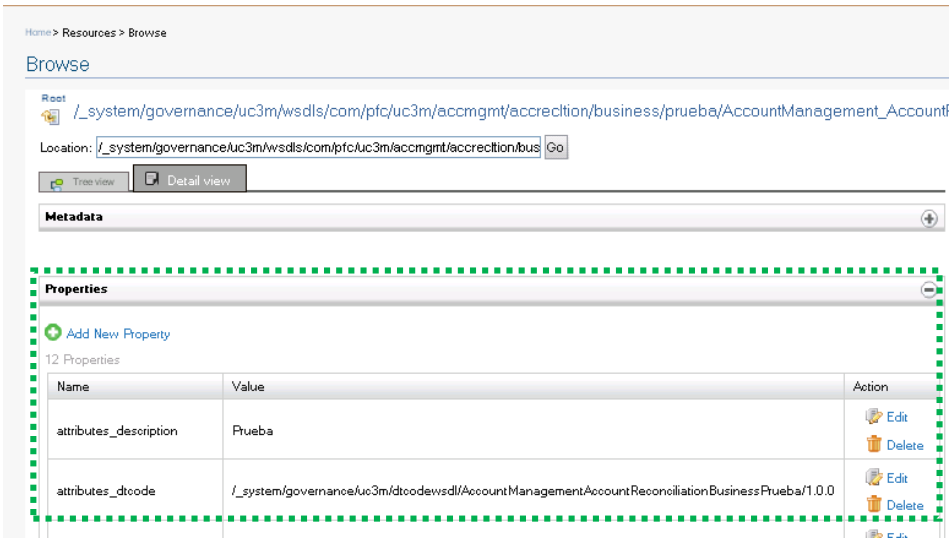


Modificación de DTCodeWSDL

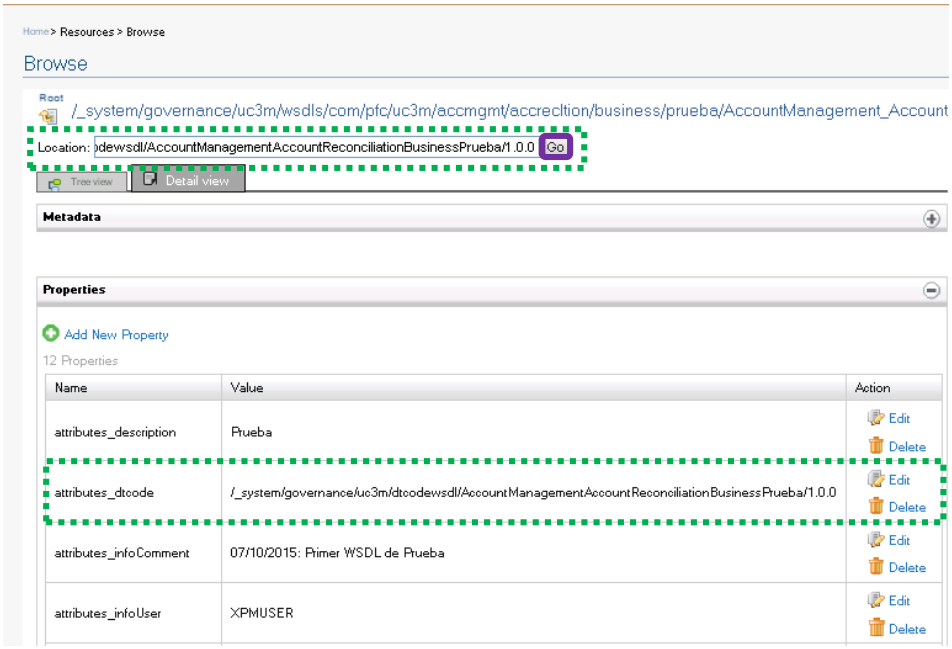
Las asociaciones de los elementos de WSO2 Governance Registry están reflejadas tanto en los artefactos del Catálogo como en el fichero DTCode, por lo tanto, también se debe modificar el DTCode para que éste incluya la asociación de los elementos.

A continuación se detallan los pasos para añadir, en el DTCode, la asociación de una Entity con un WSDL.

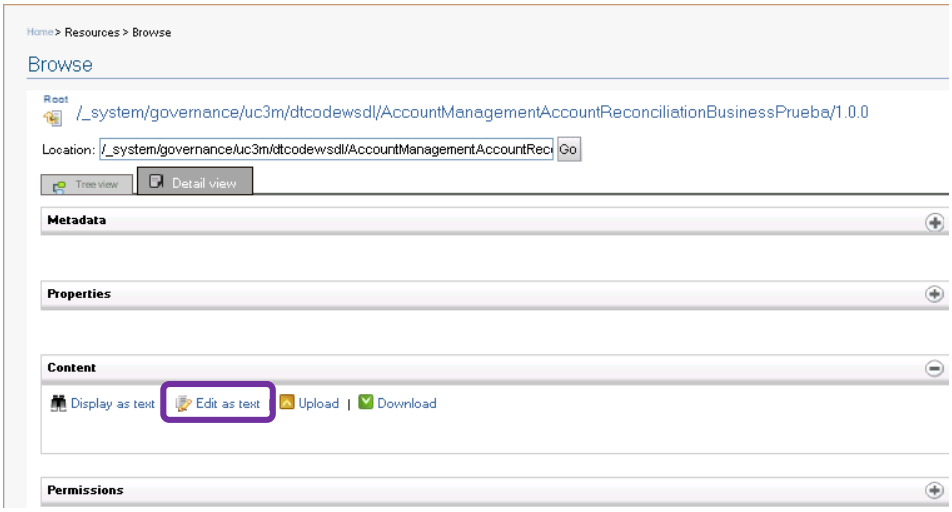
1. Desplegar la tabla ***Properties***.



2. Copiar la ruta de ***attributes_dtcode***, pegarla en ***Location*** y presionar ***Go***.



3. Presionar ***Edit as a text***.



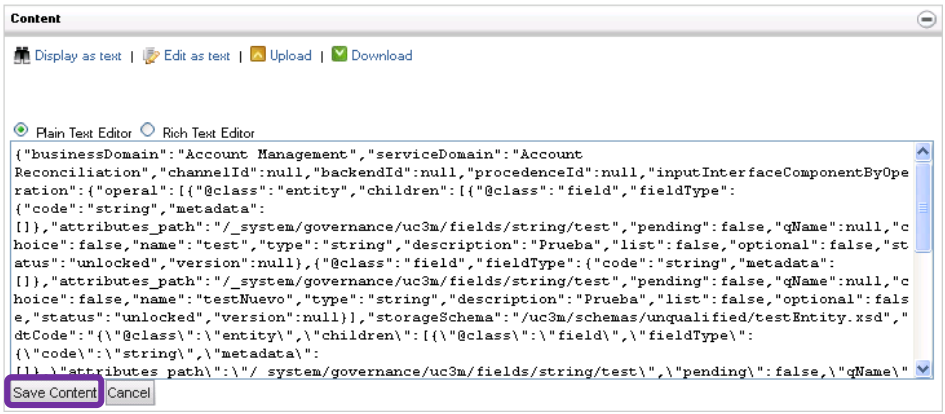
4. Añadir la Serialización DTCode de la Entity en el DTCode del WSDL.

   Ejemplo de Serialización de Entity:


```
"storageSchema":"/uc3m/schemas/unqualified/PromotionResult.xsd",
"dtCode":"{\"@class\":\"entity\",\"children\":[{\"@class\":\"field\",\"fieldTy
pe\":{\"code\":\"string\",\"metadata\":[{\"code\":\"facets_minLength\",\"type\
":\"integer\",\"label\":\"Minimum length\"
,\"stringValue\":\"\",\"defaultValue\":\"\",\"intValue\":8,\"booleanValue\":fa
lse,\"metadataType\":\"
INT\"}]},\"attributes_path\":\"/_system/governance/uc3m/fields/string/NIF\",\"
name\":\"nif\",\"type\"
:\"string\",\"description\":\" \",\"list\":false,\"optional\":false,\"status\"
:\"unlocked\",\"version\":null},
{\"@class\":\"field\",\"fieldType\":{\"code\":\"string\",\"metadata\":[]},\"at
tributes_path\":
\"/_system/governance/uc3m/fields/string/codEnt\",\"name\":\"codEnt\",\"type\"
:\"string\",
\"description\":\" \",\"list\":false,\"optional\":false,\"status\":\"unlocked\
",\"version\":null},{
\"@class\":\"field\",\"fieldType\":{\"code\":\"string\",\"metadata\":[]},\"att
ributes_path\":\"/_system/governance/uc3m/fields/string/description\",\"name\"
:\"description\",\"type\":\"string\",\"description\"
:\"Descripcion del campo\",\"list\":false,\"optional\":false,\"status\":\"unlo
cked\",\"version\":n
ull}],\"storageSchema\":null,\"dtCode\":null,\"nature\":null,\"name\":null,\"t
ype\":null,\"description\":null,
\"list\":false,\"optional\":false,\"version\":null,\"attributes_path\":null}",
"nature":"Business",
"name":"promotionResult",
"type":"PromotionResult",
"description":" ",
"list":false,
"optional":false,
"version":"1.2.0",
"attributes_path":"/_system/governance/uc3m/entities/Business/Promo
tionResult/1.2.0"
```


NOTA: El código anterior corresponde con la definición JSON del Entity y se debe añadir bajo el array JSON de la operación del WSDL.

5. Una vez se ha añadido la Entity presionar ***Save Content***.



B.2.2 Bloqueo y Desbloqueo de Elementos

El bloqueo de los elementos del Catálogo de Referencia es necesario para evitar que más de un desarrollador modifique el mismo elemento a la vez y se pierdan los cambios realizados. Durante la creación o edición de un elemento del Catálogo éste puede quedarse bloqueado, por lo tanto, para que se pueda trabajar en él es necesario desbloquearlo desde la consola de WSO2 Governance Registry.

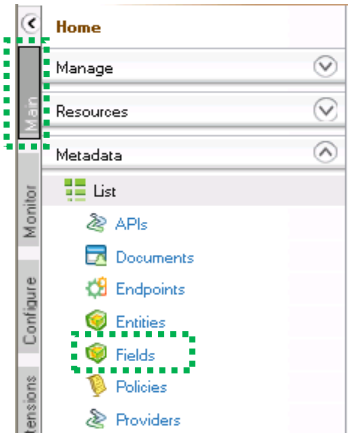
Ya que el método de desbloqueo es diferente para cada tipo de elemento, se describirán los pasos a seguir para bloquear o desbloquear:

- Elementos *Field*
- Elementos *Entity*
- Elementos WSDL

B.2.2.1 Elementos Field

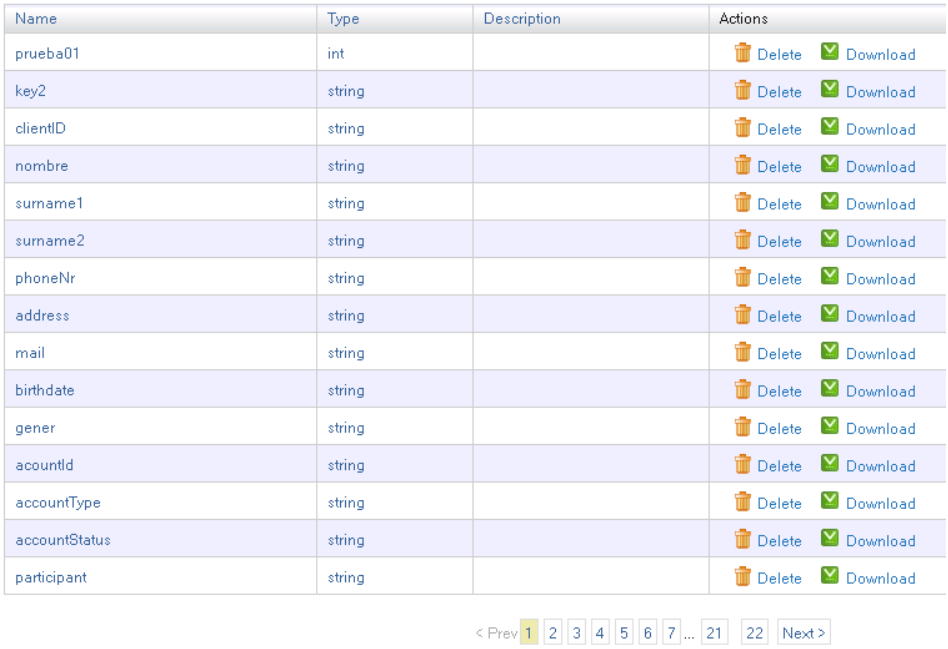
Los pasos a seguir para bloquear y desbloquear elementos de tipo Field del Catálogo de Referencia son los siguientes:

1. Presionar sobre ***Main*** **>** ***Fields***.



2. Para encontrar el Field que se quiere bloquear o desbloquear hay dos opciones:

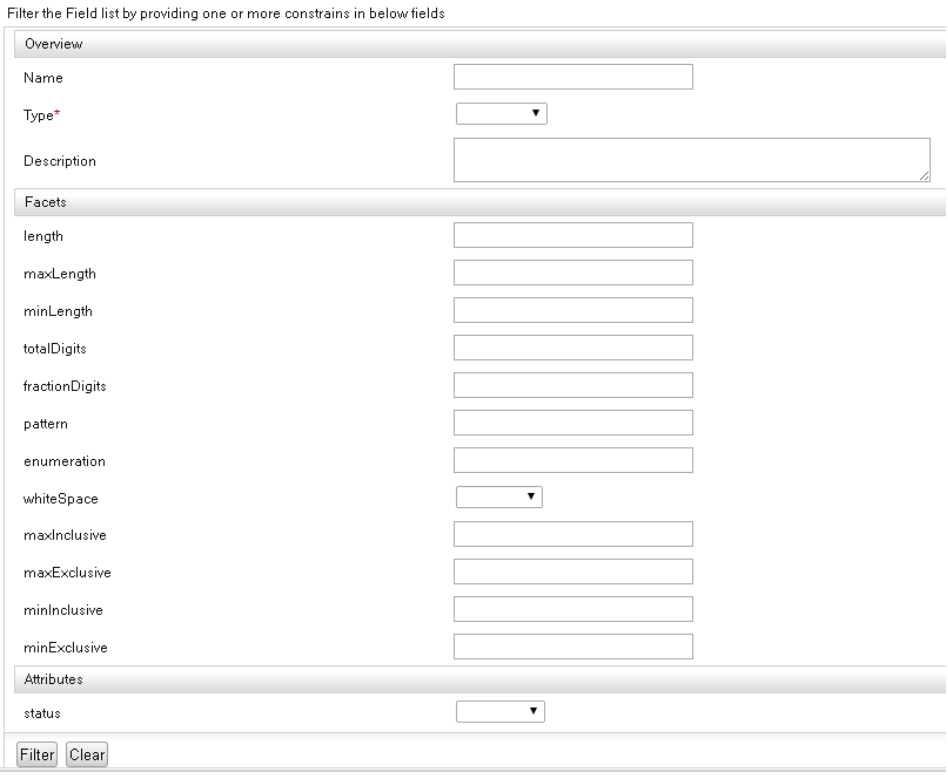
   a. Navegar por el listado de Fields hasta encontrar el que se busca.



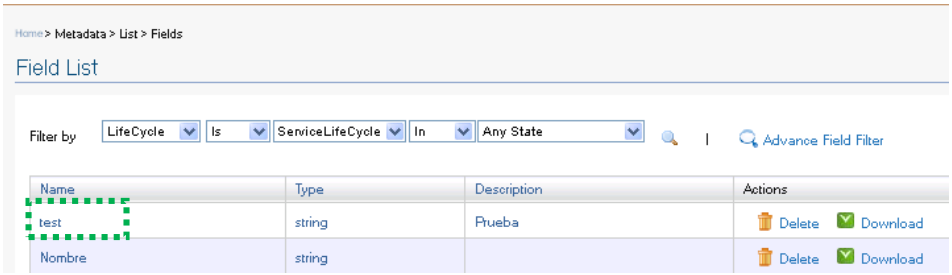
Name	Type	Description	Actions
prueba01	int		Delete Download
key2	string		Delete Download
clientID	string		Delete Download
nombre	string		Delete Download
surname1	string		Delete Download
surname2	string		Delete Download
phoneNr	string		Delete Download
address	string		Delete Download
mail	string		Delete Download
birthdate	string		Delete Download
gener	string		Delete Download
acountId	string		Delete Download
accountType	string		Delete Download
accountStatus	string		Delete Download
participant	string		Delete Download

< Prev 1 2 3 4 5 6 7 ... 21 22 Next >

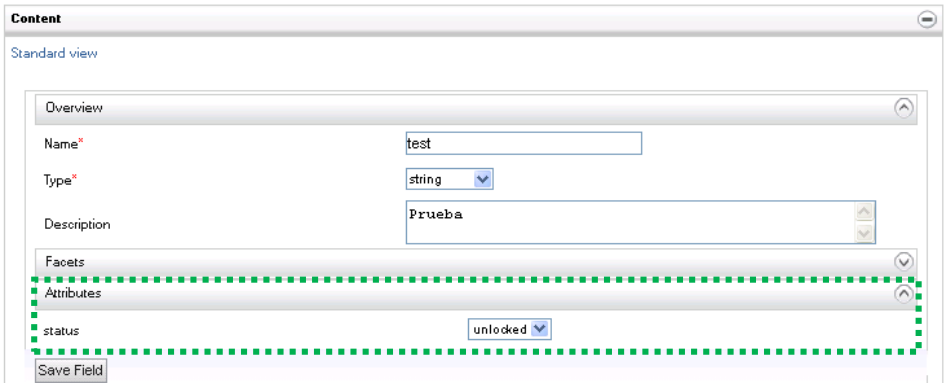
   b. Utilizar la búsqueda avanzada de la consola. Es obligatorio insertar el tipo del Field.



3. Una vez se ha encontrado el ***Field*** a bloquear o desbloquear, presionar sobre él.

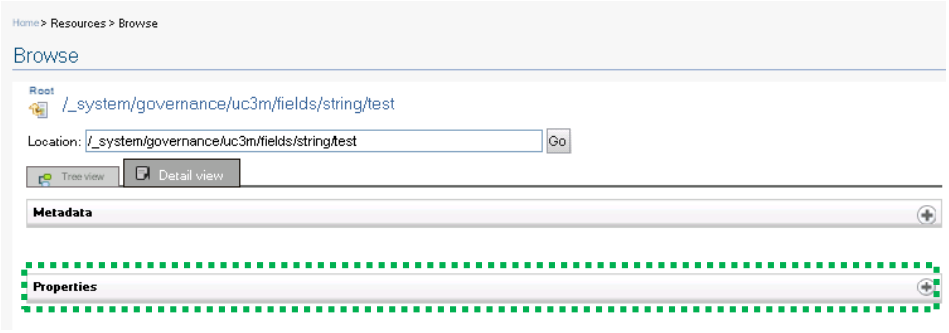


4. En la tabla ***Content***, bajo ***Attributes*** seleccionar el valor de status *locked* (bloqueado) o *unlocked* (desbloqueado).

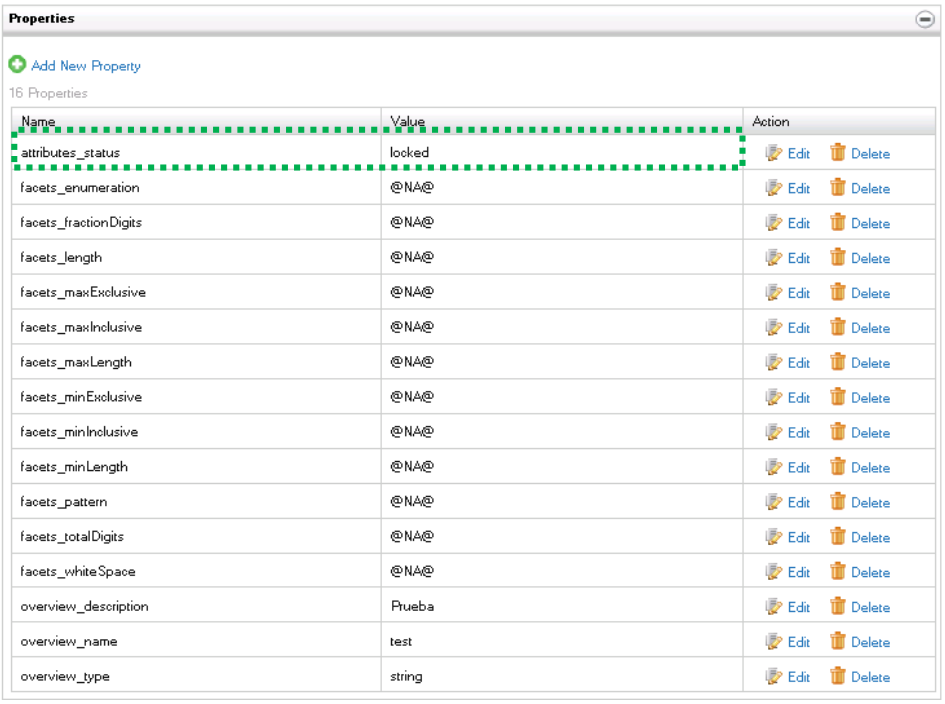


5. Presionar ***Save Field*** para cambiar el estado del Campo.

6. Desplegar la tabla ***Properties***.



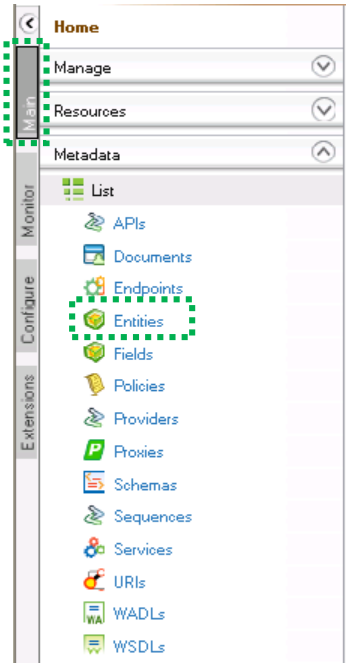
7. Comprobar que la propiedad ***attribute_status*** tiene el valor que le hemos asignado.



B.2.2.2 Elementos Entity

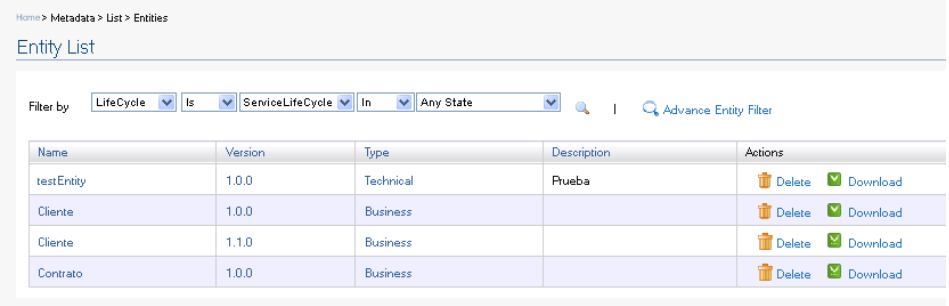
Los pasos a seguir para bloquear y desbloquear elementos de tipo Entity del Catálogo de Referencia son los siguientes:

1. Presionar sobre ***Main*** **>** ***Entities***.

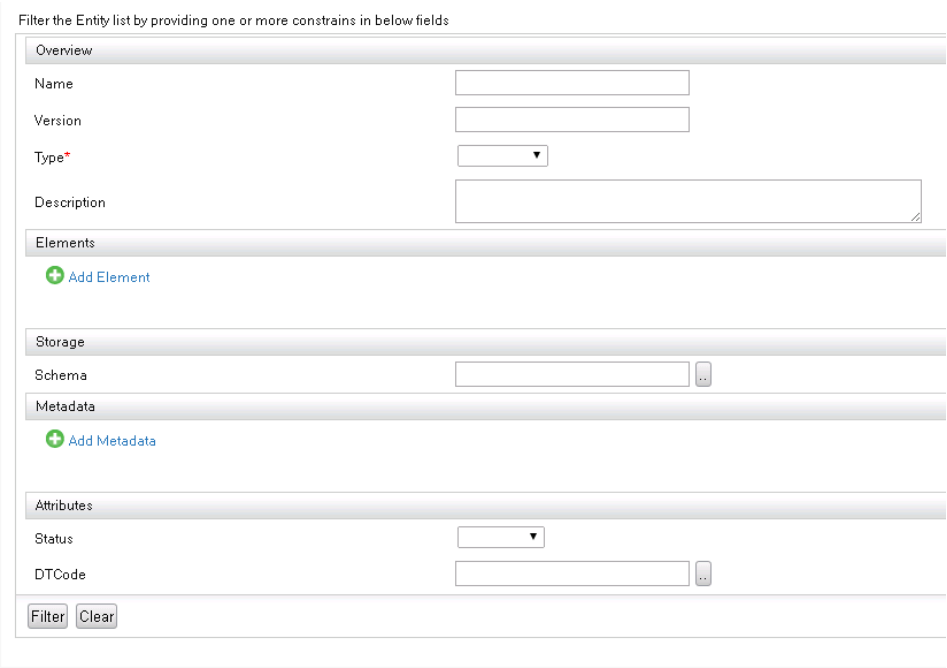


2. Para encontrar la Entity que se quiere bloquear o desbloquear hay dos opciones:

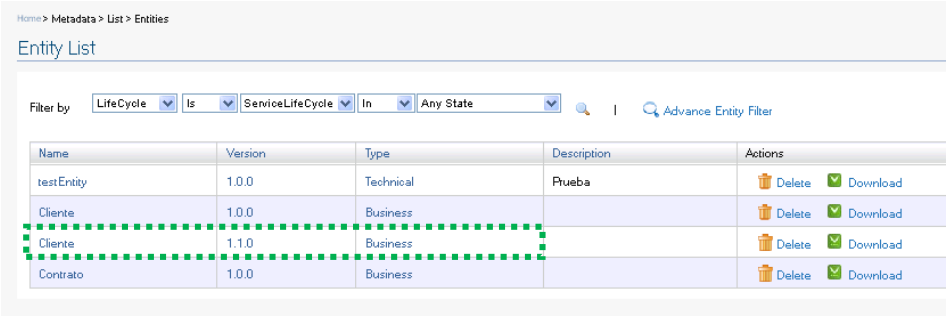
   a. Navegar por el listado de Entities hasta encontrar la que se busca.



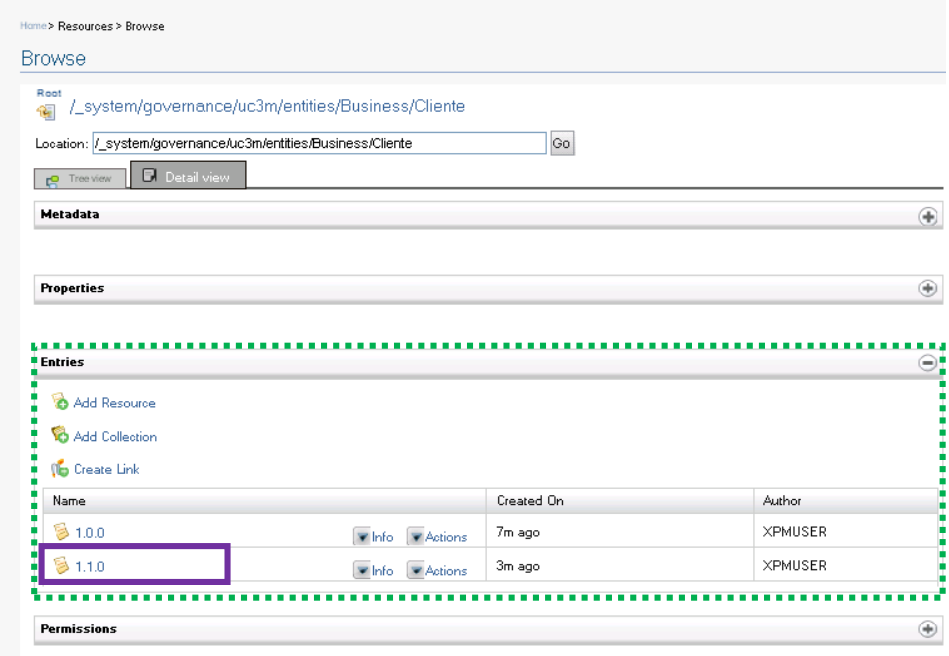
   b. Utilizar la búsqueda avanzada de la consola. Es obligatorio insertar el tipo del Entity.



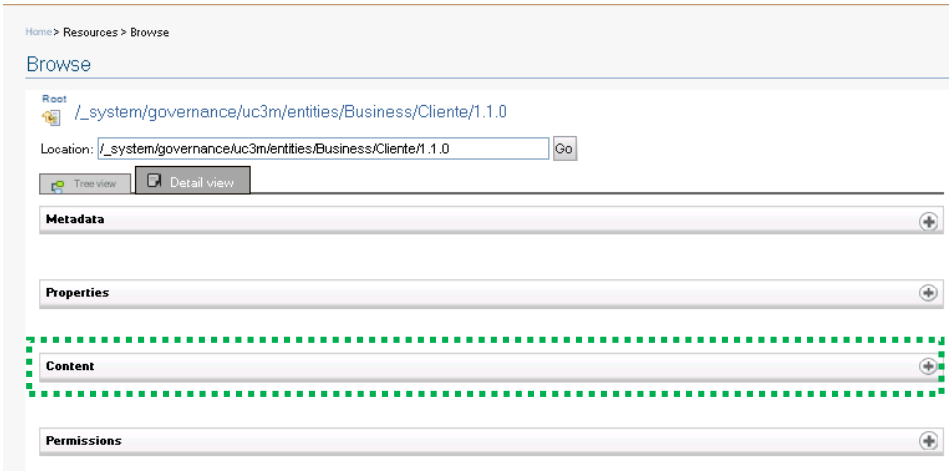
3. Una vez se ha encontrado la ***Entity*** a bloquear o desbloquear, presionar sobre ella.



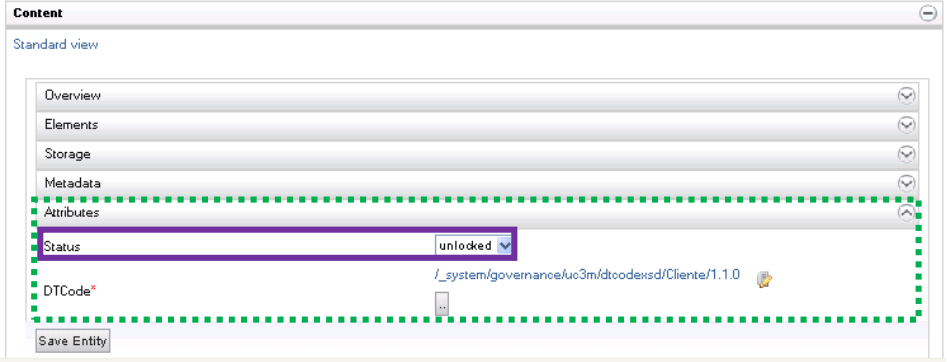
4. En el apartado ***Entries*** presionar sobre la ***última versión*** de la Entity.



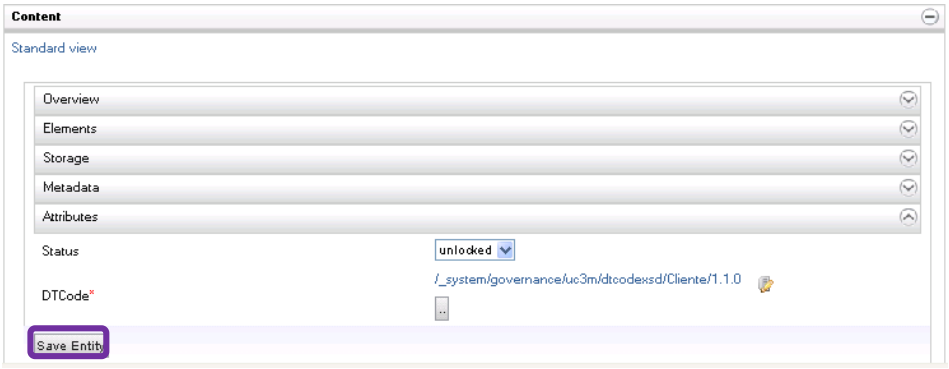
5. Desplegar ***Content***.



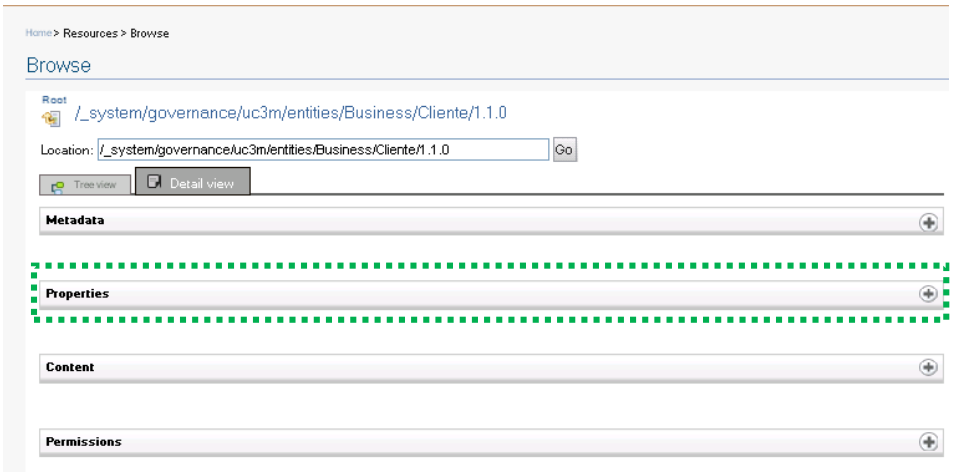
6. En ***Attributes*** modificar el ***Status*** a *locked* (bloqueado) o *unlocked* (desbloqueado).



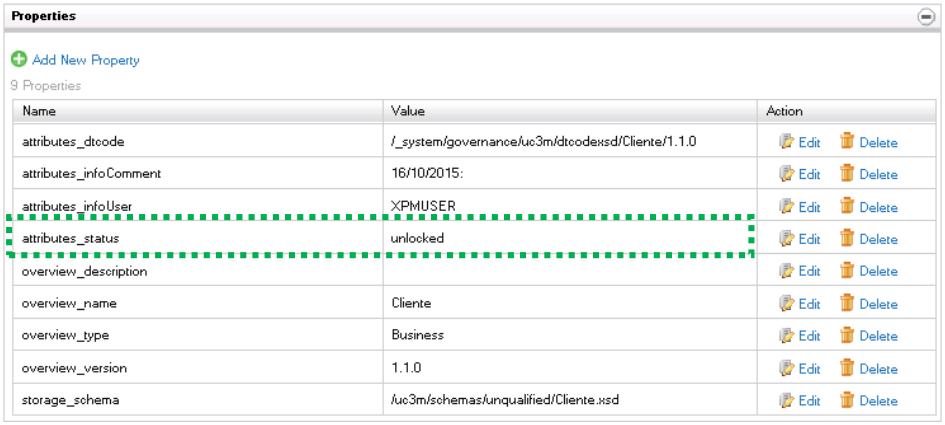
7. Presionar ***Save Entity***.



8. Desplegar la tabla ***Properties***.



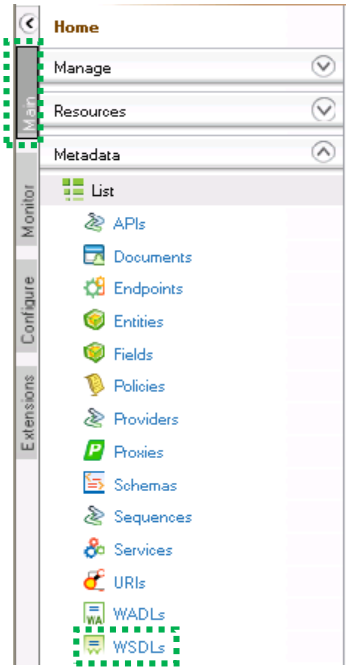
9. Comprobar que la propiedad ***attribute_status*** tiene el valor que le hemos asignado.



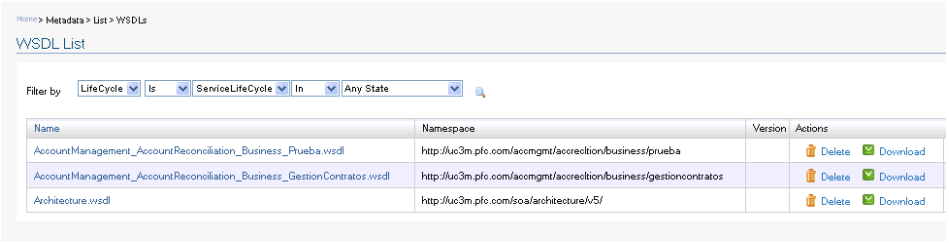
B.2.2.3 Elementos WSDL

Los pasos a seguir para bloquear y desbloquear elementos de tipo WSDL del Catálogo de Referencia son los siguientes:

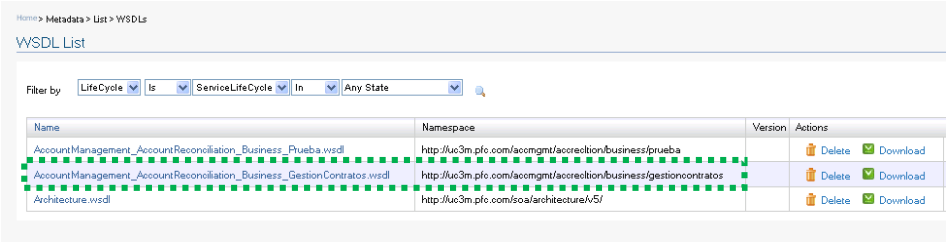
1. Presionar sobre ***Main*** **>** ***WSDLs***.



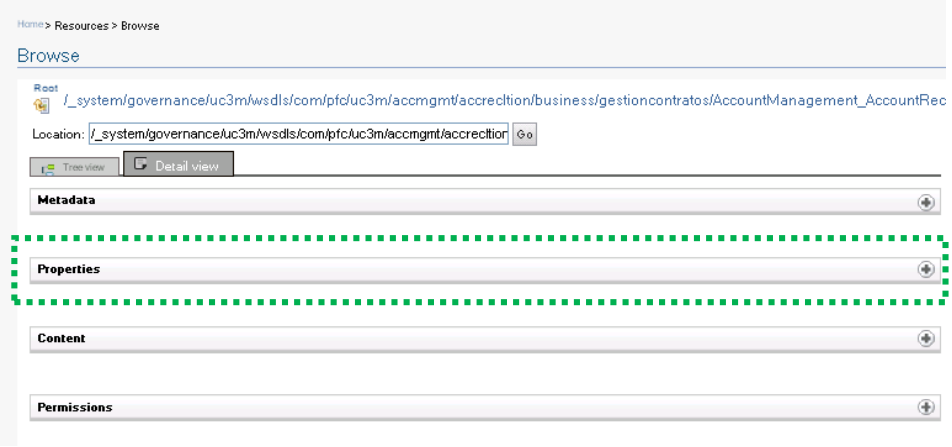
2. Para encontrar el WSDL que se quiere bloquear o desbloquear hay que navegar por el listado de WSDLs hasta encontrar el que se busca.



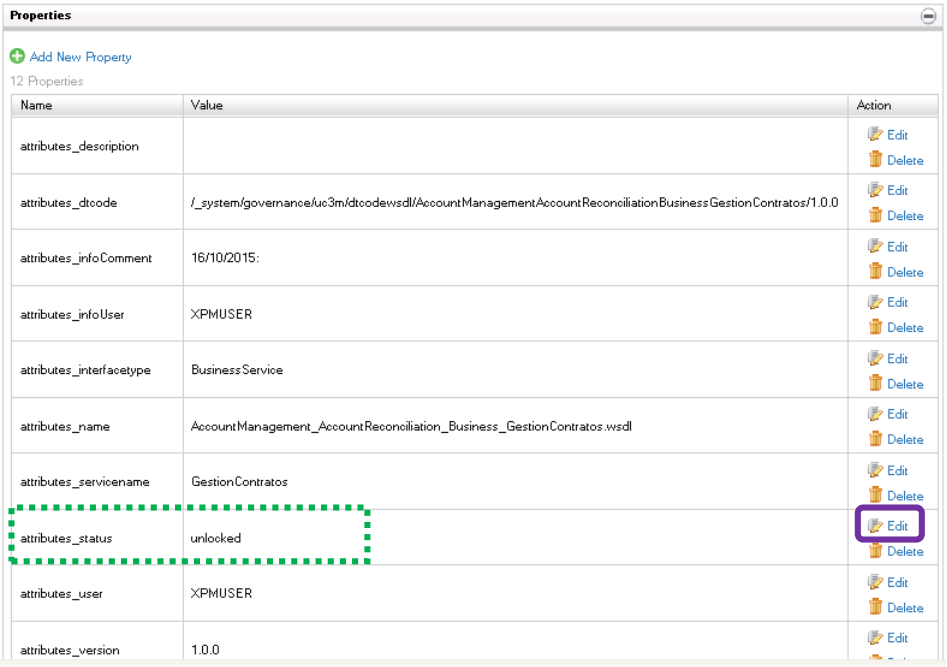
3. Una vez se ha encontrado el ***WSDL*** a bloquear o desbloquear, presionar sobre él.



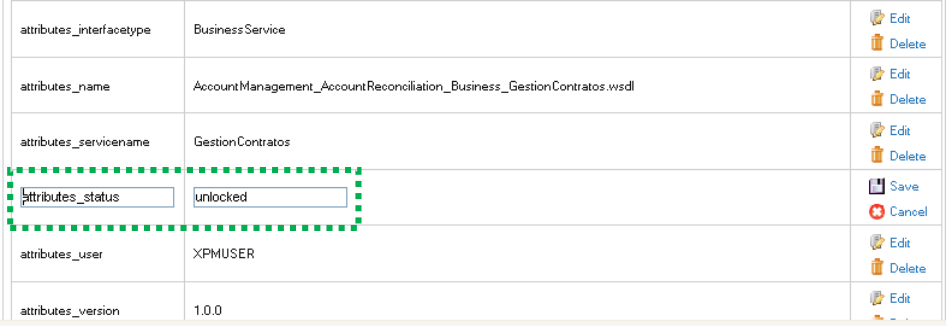
4. Deplegar ***Properties***.



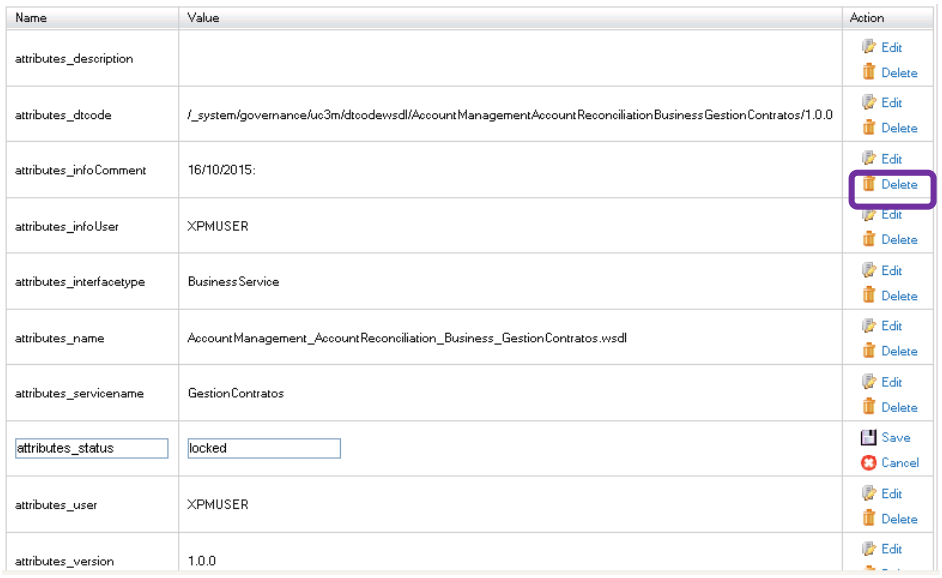
5. Buscar la propiedad ***attributes_status*** y presionar ***Edit***.



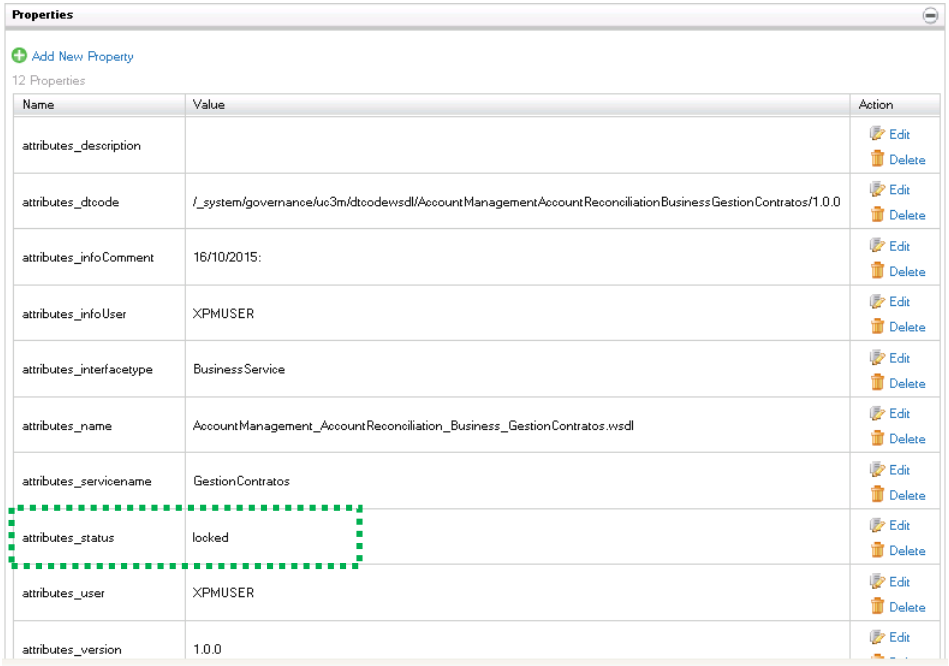
6. Modificar el **Value** a **locked** (bloqueado) o **unlocked** (desbloqueado) en función del estado en que se quiera el WSDL.



7. Una vez se ha modificado el ***Value*** presionar ***Save***.



8. Finalmente, el valor de ***attributes_status*** habrá cambiado.



B.2.3 Borrado de Elementos

Los artefactos del Catálogo de Referencia pueden ser borrados por el usuario desde la consola de WSO2 Governance Registry. Los artefactos del Catálogo de Referencia están formados por diferentes elementos de Registry en función de su tipo. Es importante saber por qué elementos está formado cada artefacto del Catálogo para que su borrado sea completo.

En la siguiente tabla se muestran los elementos de Registry que conforman cada tipo de artefacto del Catálogo de Referencia.

Artefacto	DTCode	Contenedor
Field	-	-
Entity	DTCodeXSD	Schema
WSDL	DTCodeWSDL	WSDL

Hay dos opciones de borrado para los elementos de WSO2 Governance Registry:

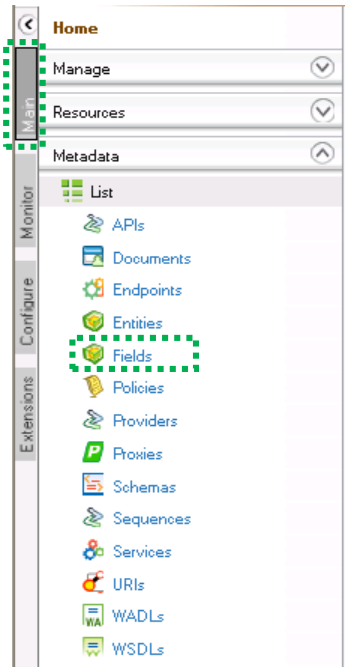
- **Borrado total**, que consiste en borrar todas las versiones de una Entidad, WSDL o Campo.
- **Borrado de versión**, que consiste en borrar una versión concreta de una Entidad o WSDL. Este caso no aplica para un artefacto de tipo Campo, ya que no están versionados.

Los siguientes capítulos detallan los pasos para borrar una versión de un artefacto Field, Entity y WSDL.

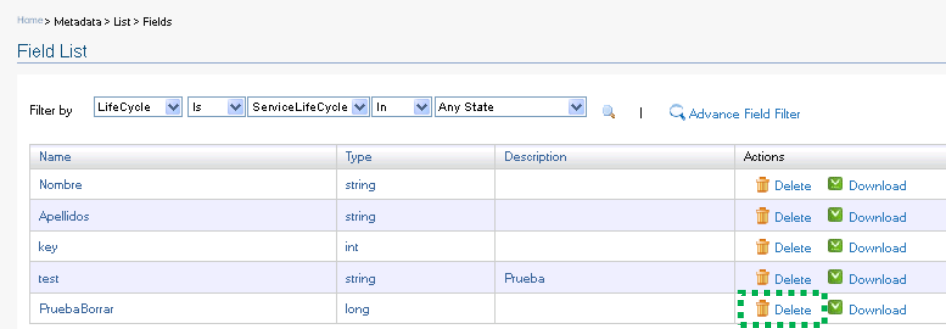
B.2.3.1 Elementos Field

Los pasos para borrar un elemento Field de WSO2 Governance Registry son los siguientes:

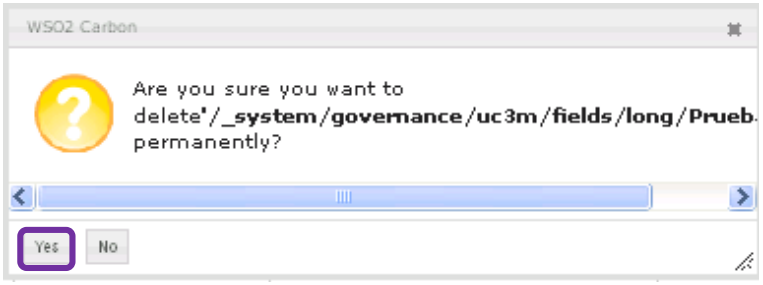
1. Presionar sobre ***Main*** **>** ***Fields***.



2. Buscar en el listado el Field que se quiere borrar y presionar ***Delete***.



3. Finalmente, presionar ***Yes*** en el diálogo de confirmación de borrado.



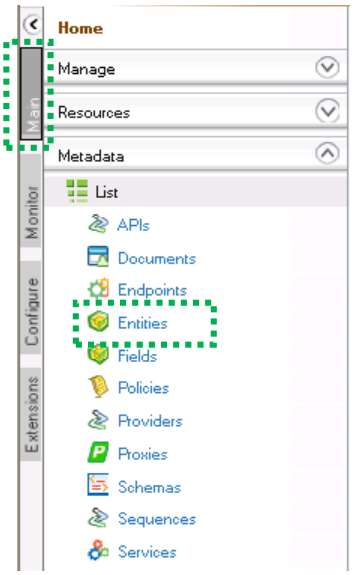
B.2.3.2 Elementos Entity

Los elementos Entity en WSO2 Governance Registry están formados por un artefacto Entity, una serialización DTCode y un schema.

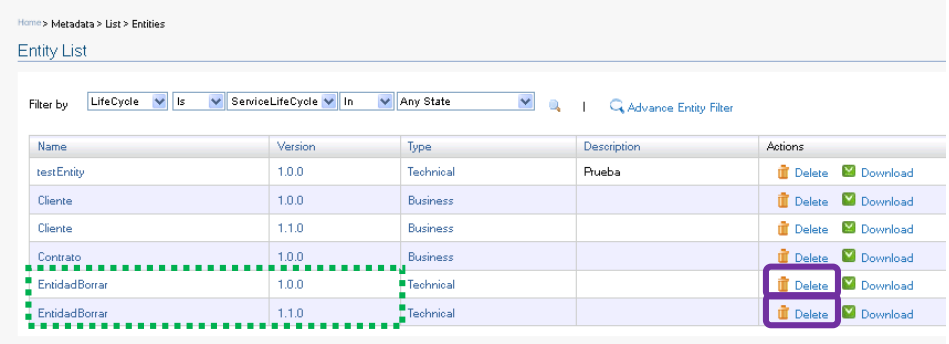
Borrado de Artefacto

Los pasos a seguir para borrar un artefacto Entity de WSO2 Governance Registry son los siguientes:

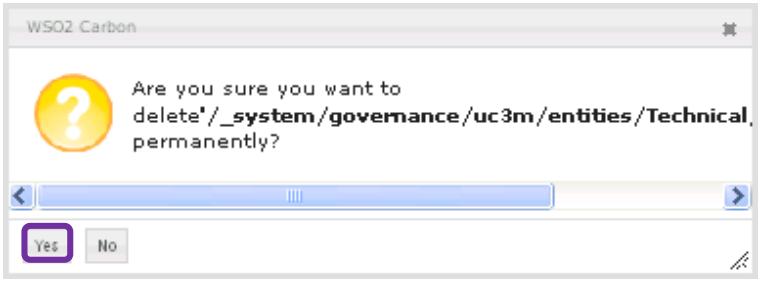
1. Presionar sobre ***Main*** **>** ***Entities***.



2. Navegar por el listado de Entities hasta encontrar la que se busca y presionar ***Delete*** sobre la versión del artefacto que se desee eliminar.



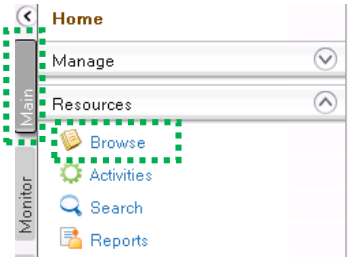
3. Finalmente, presionar ***Yes*** en el diálogo de confirmación de borrado.



Borrado de DTCode

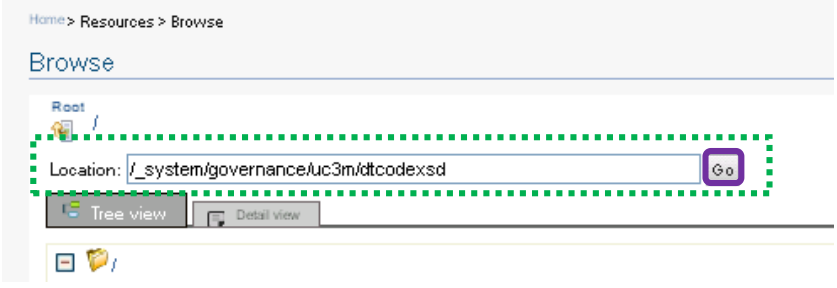
Los pasos a seguir para borrar una versión de la serialización DTCode de WSO2 Governance Registry son los siguientes:

1. Presionar sobre ***Main*** **>** ***Browse***.

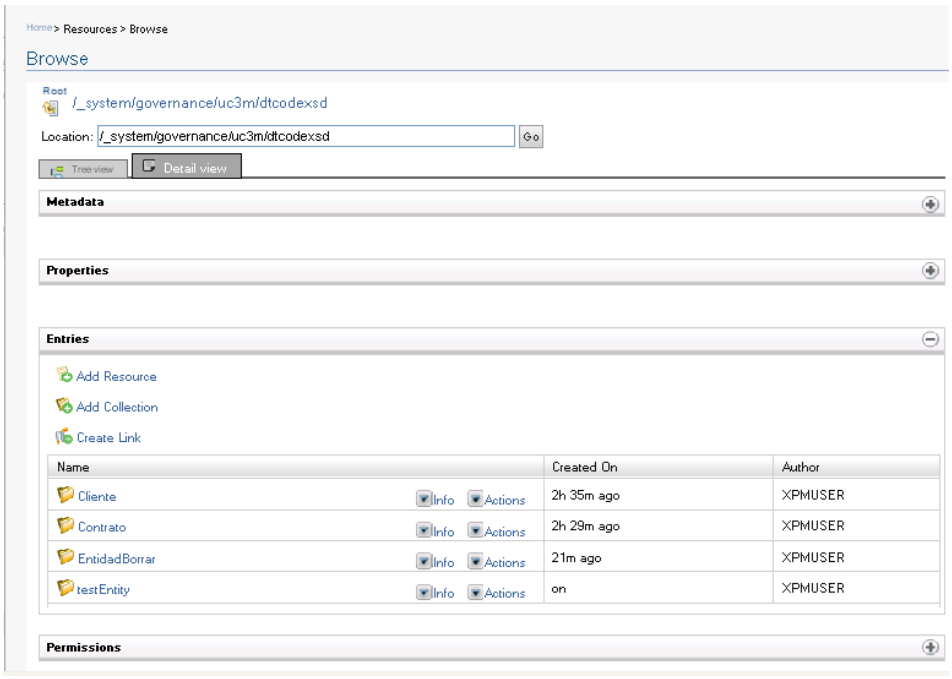


2. En ***Location*** pegar la siguiente ruta y presionar ***Go***:

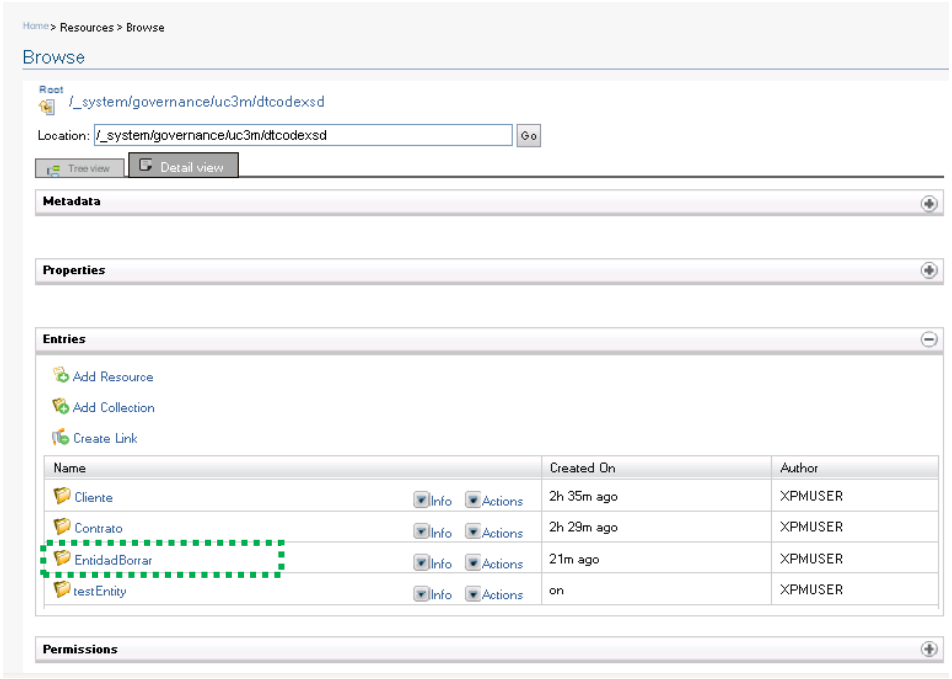
/_system/governance/uc3m/dtcodexsd



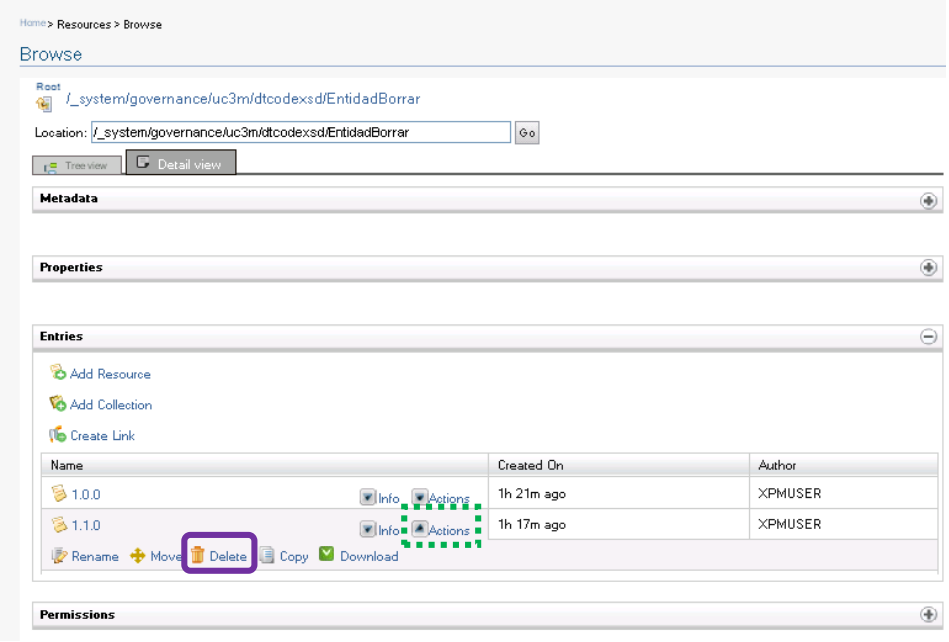
3. En la tabla **Entries**, navegar por los ficheros hasta encontrar el directorio que contiene las versiones del DTCode correspondiente al artefacto Entity.



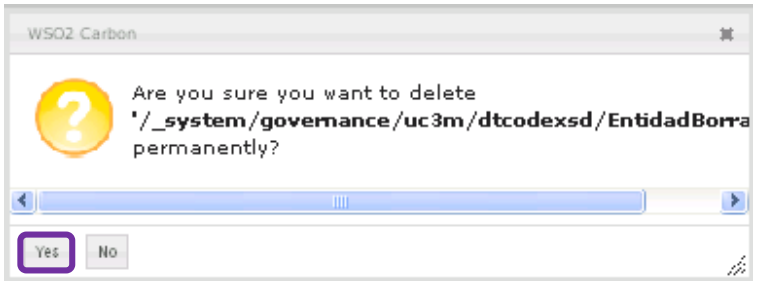
4. Una vez encontrado el directorio de DTCodes presionar sobre él.



5. Desplegar ***Actions*** y presionar ***Delete*** de la versión del DTCode que se desea borrar.



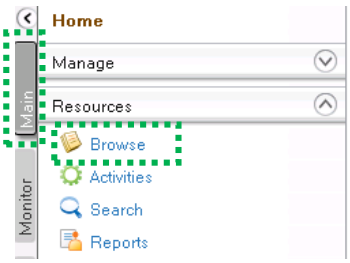
6. Finalmente, presionar ***Yes*** en el diálogo de confirmación de borrado.



Borrado de Schema

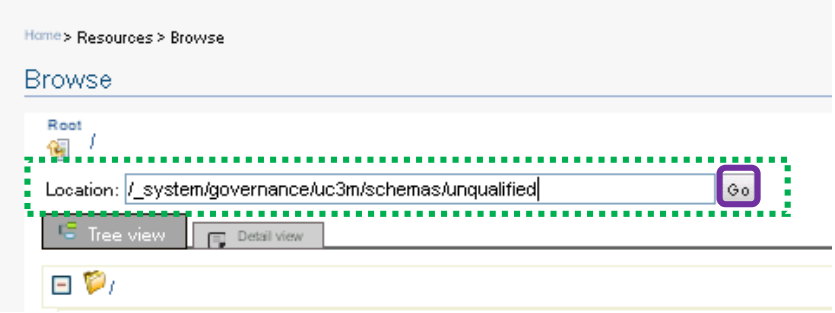
Los pasos a seguir para borrar el schema XSD de la Entity de WSO2 Governance Registry son los siguientes:

1. Presionar sobre ***Main*** **>** ***Browse***.



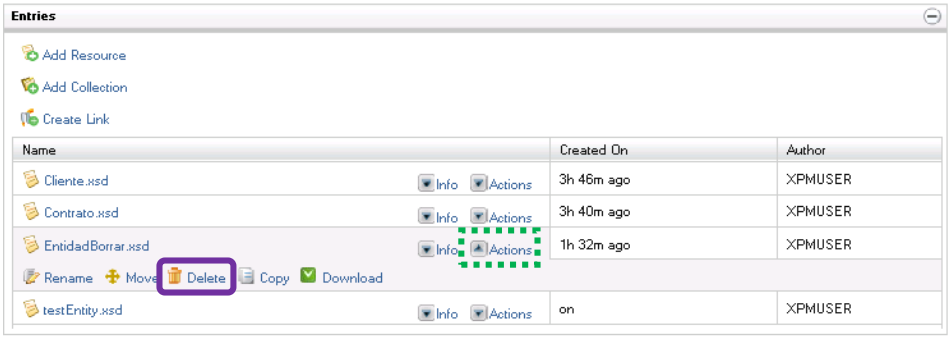
2. En ***Location*** pegar la siguiente ruta y presionar ***Go***:

/_system/governance/uc3m/schemas/unqualified/

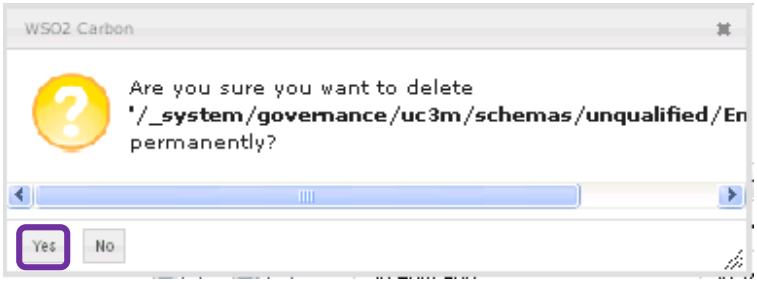


3. En la tabla ***Entries***, navegar por los ficheros hasta encontrar el schema correspondiente al artefacto Entity.

4. Una vez encontrado el schema presionar ***Actions*** y ***Delete***.



5. Finalmente, presionar ***Yes*** en el diálogo de confirmación de borrado.



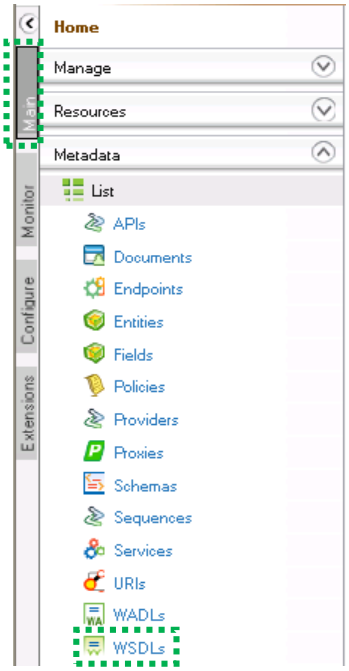
B.2.3.3 Elementos WSDL

Los elementos WSDL en WSO2 Governance Registry están formados por un artefacto WSDL y una serialización DTCode.

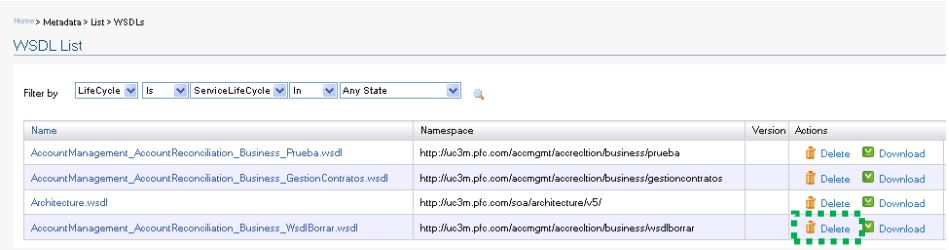
Borrado de Artefacto

Los pasos a seguir para borrar el artefacto WSDL del Catálogo de Referencia son los siguientes:

1. Presionar sobre ***Main*** **>** ***WSDLs***.



2. Navegar por el listado de WSDLs hasta encontrar el que se quiere borrar y presionar ***Delete***.

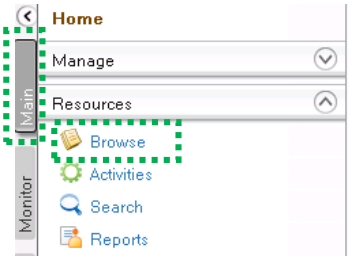


3. Finalmente, presionar ***Yes*** en el diálogo de confirmación de borrado.

Borrado de DTCode

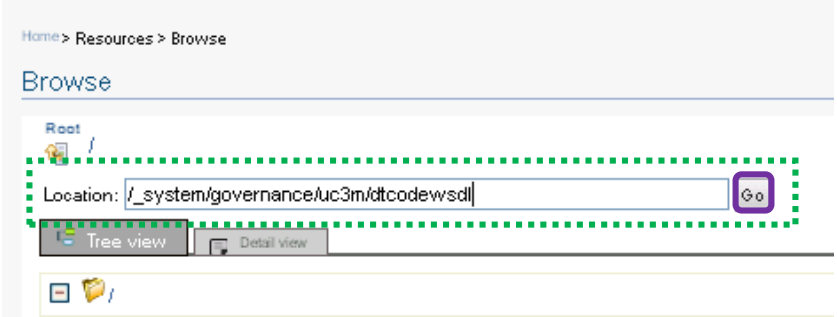
Los pasos a seguir para borrar la serialización DTCode de WSO2 Governance Registry son los siguientes:

1. Presionar sobre ***Main*** **>** ***Browse***.

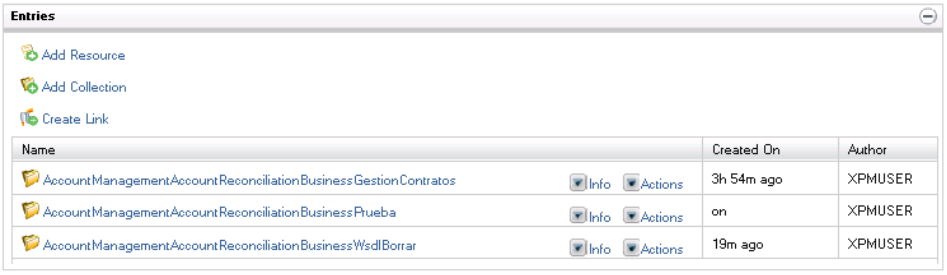


2. En ***Location*** pegar la siguiente ruta y presionar ***Go***:

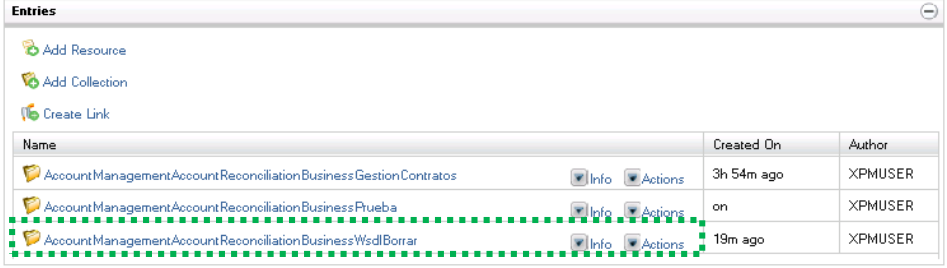
/_system/governance/uc3m/dtcodewsdl



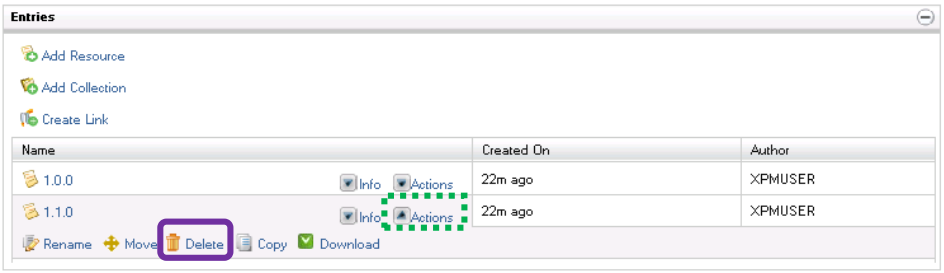
3. En la tabla ***Entries***, navegar por los ficheros hasta encontrar el direcotrio de DTCodes correspondientes al artefacto WSDL.



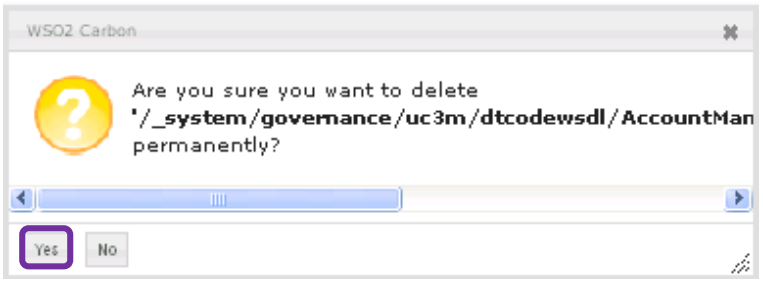
4. Una vez encontrado el directorio de DTCodes presionar sobre él.



5. Desplegar ***Actions*** y presionar ***Delete*** de la versión del DTCode que se desea borrar.



6. Finalmente, presionar ***Yes*** en el diálogo de confirmación de borrado.



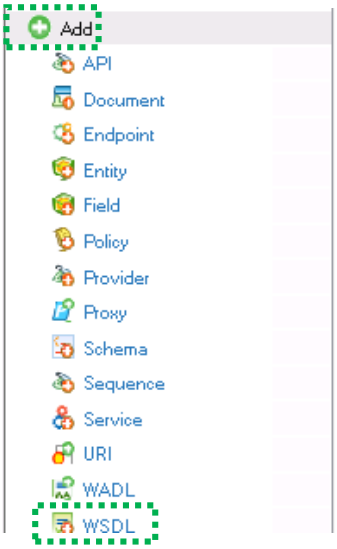
B.3 Publicación de Recursos WSDL

La publicación de los recursos WSDL en WSO2 Governance Registry permite añadir elementos WSDL en el Catálogo de Referencia. Por defecto, al publicar recursos no son accesibles por todo el mundo y es necesario loguearse para poder acceder a ellos.

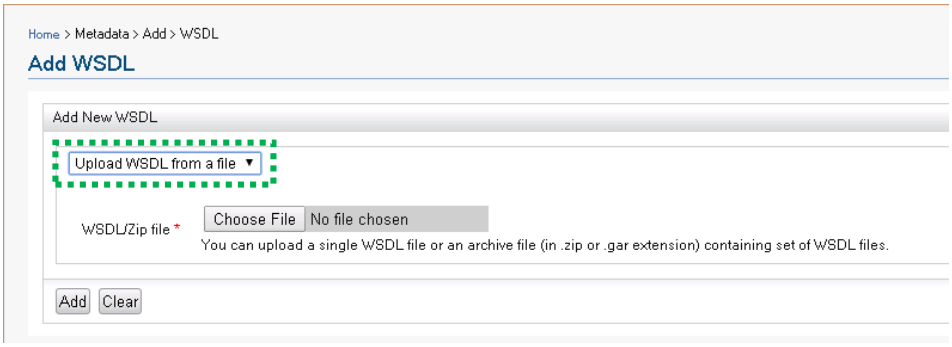
NOTA: Si se añaden manualmente los WSDLs al Catálogo de Referencia no se garantiza su explotación desde las herramientas de desarrollo.

A continuación se detallan los pasos para publicar un WSDL en el Catálogo de Referencia.

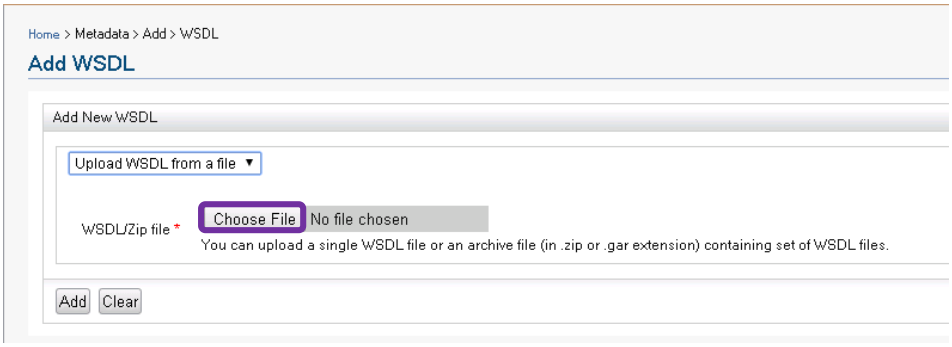
1. Presionar ***Add*** **>** ***WSDL***.



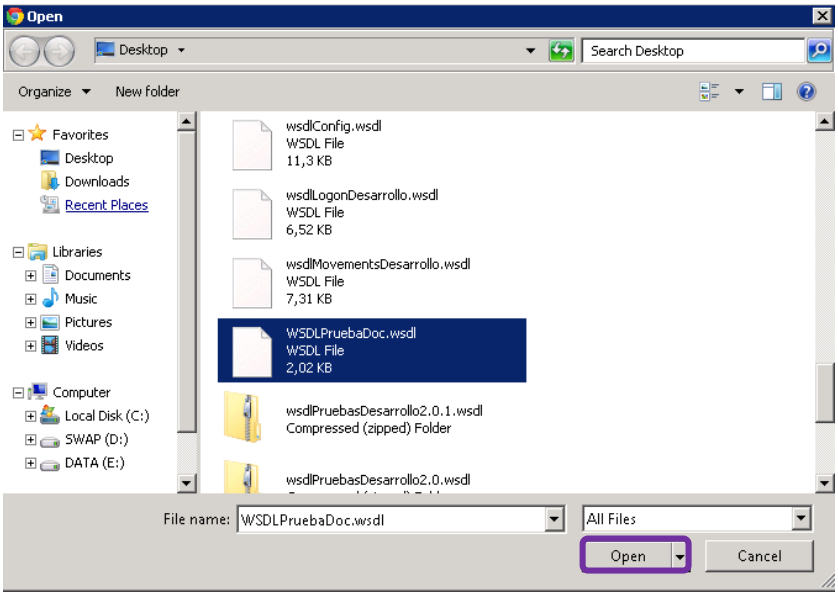
2. Seleccionar ***Upload WSDL from a file***.



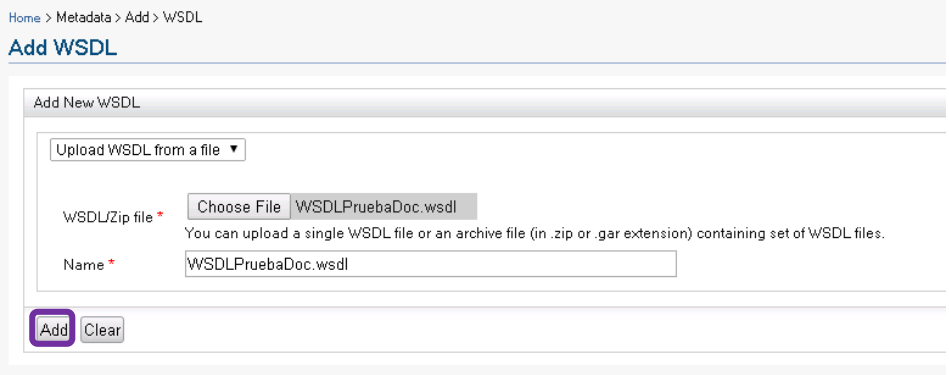
3. Presionar ***Choose File***.



4. Navegar por el Sistema de ficheros para seleccionar el WSDL que se desea cargar y presionar ***Open***.



5. Presionar ***Add***.

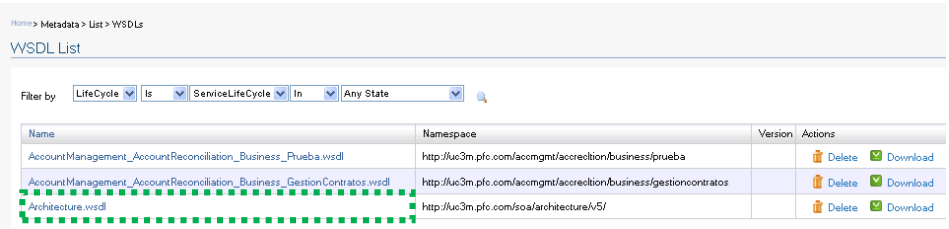


B.3.1 Accesibilidad de los Recursos

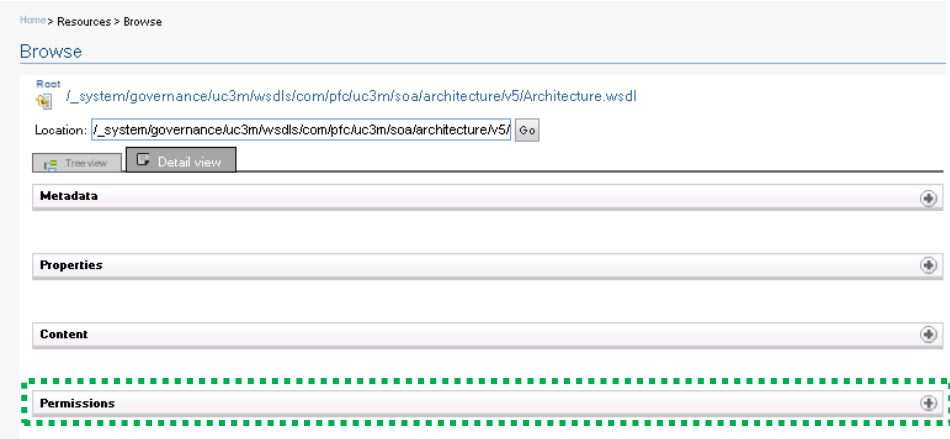
En algunos casos, es deseable que un WSDL publicado en Registry sea accesible por todo el mundo; de este modo, en tiempo de resolución de dependencias (cuando se publica un WSDL que realiza un *import* de otro WSDL) no habrá problemas de acceso de lectura.

A continuación se detallan los pasos a seguir para que el WSDL sea accesible por todos.

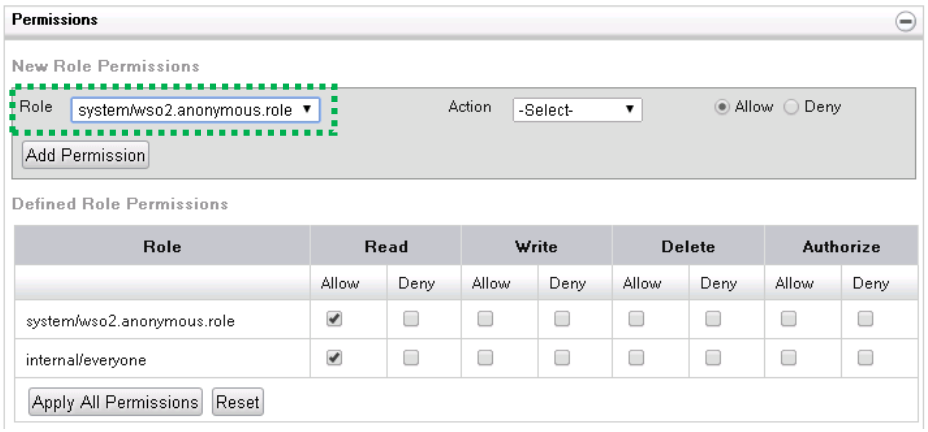
1. En el listado de WSDLs, presionar sobre el ***WSDL*** que vaya a tener permiso de lectura público.



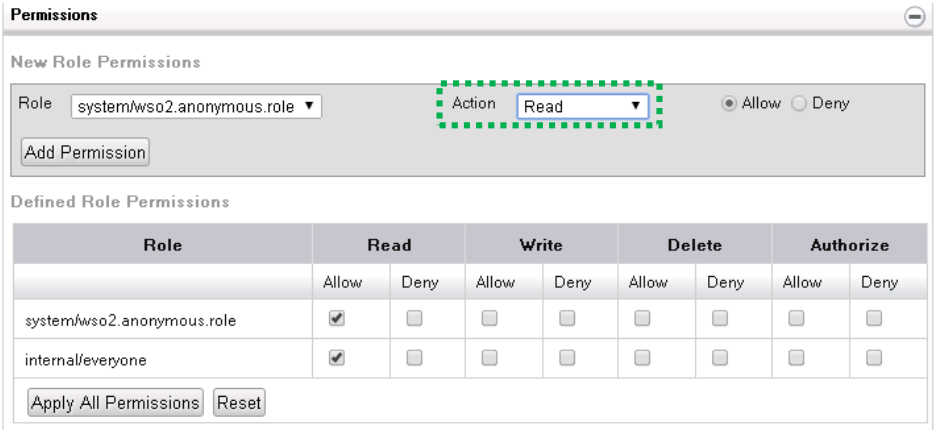
2. Desplegar la tabla ***Permissions***.



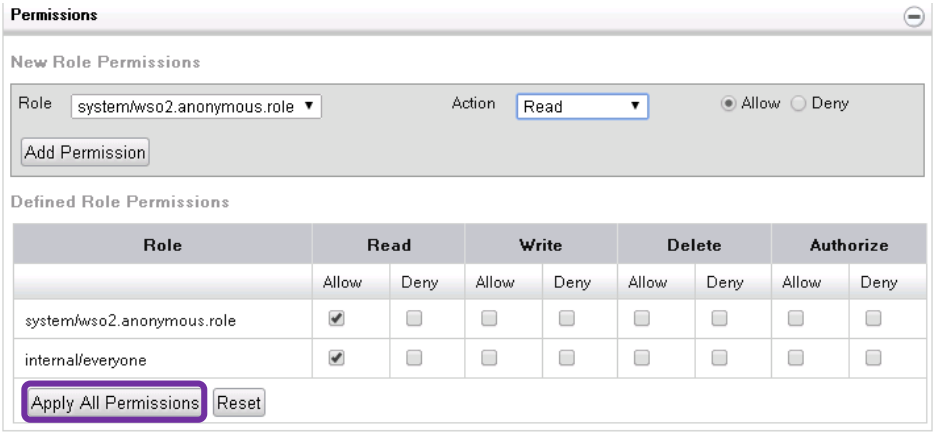
3. En ***Role*** seleccionar ***system/wso2.anonymous.role***.



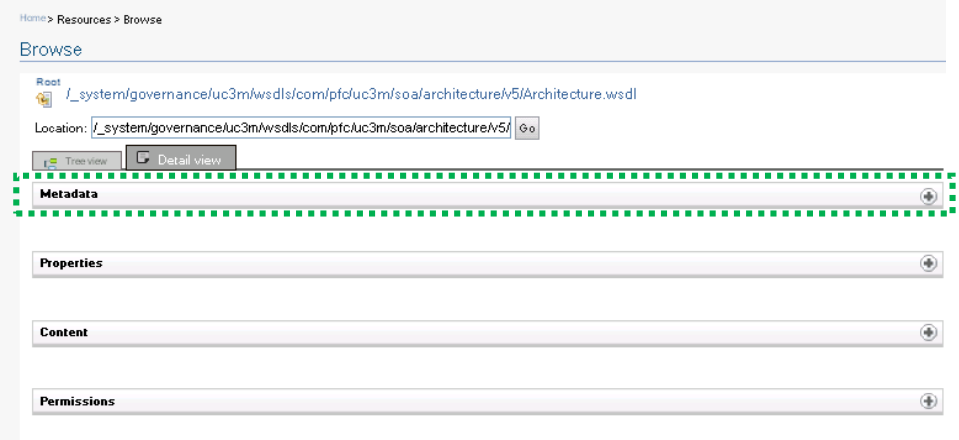
4. En ***Action*** seleccionar la opción ***Read***.



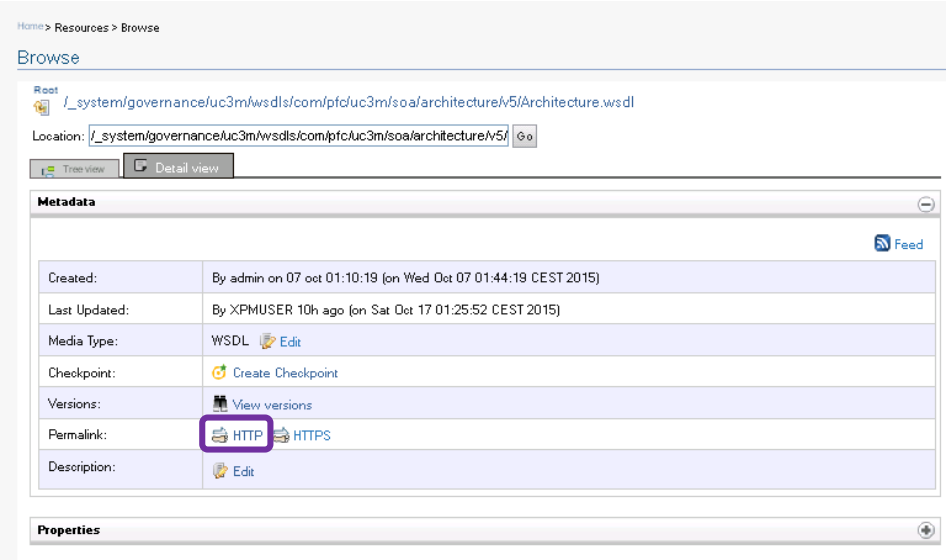
5. Presionar ***Apply All Permissions*.**



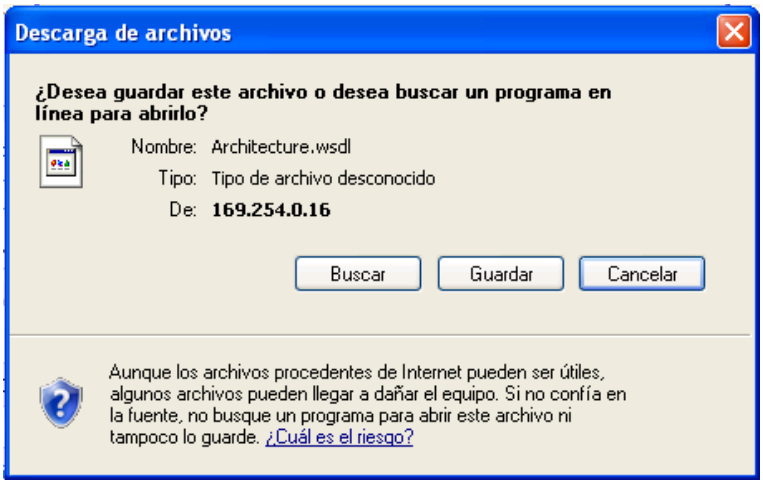
6. Para verificar que el acceso al WSDL es público, desplegar la tabla ***Metadata***.



7. Presionar ***HTTP***.



8. El WSDL se descargará sin pedir datos de acceso.



Anexo C.

Guía de Uso de IDE Utils

El catálogo de referencia es el repositorio centralizado de artefactos que alberga los atributos, las entidades lógicas (de negocio y técnicas) y las interfaces de los servicios desarrollados sobre la Arquitectura Digital.

El entorno local de desarrollo pone al servicio del usuario la herramienta *IDE Utils*, integrada con Eclipse, que permite mantener el catálogo y definir las interfaces necesarias para la creación e invocación de servicios aplicativos.

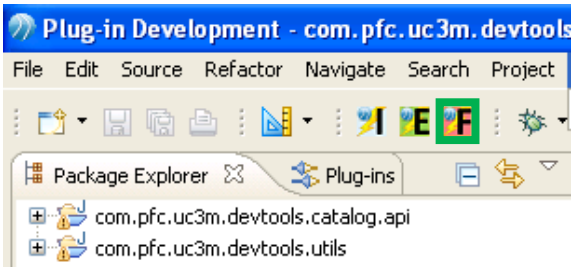
El objetivo de este documento es describir el procedimiento a seguir para modelar servicios bajo los estándares previamente definidos; a continuación, se detalla el funcionamiento de la utilidad para cada uno de los elementos contenidos en el catálogo de referencia.

C.1 Campos en el Catálogo de Referencia – Field Generator

Esta ampliación de funcionalidad permite realizar el mantenimiento (búsqueda, creación, edición) de los campos del Catálogo de Referencia desde el Eclipse del entorno local de desarrollo.

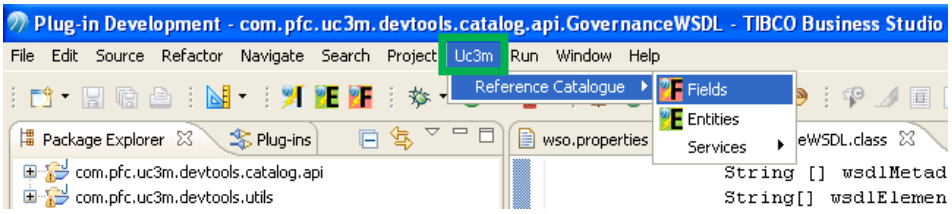
Para abrir la herramienta *Field Generator*, que permite este mantenimiento de campos del Catálogo de Referencia, hay dos opciones:

a. Presionar el botón ***Fields*** de la barra de herramientas.

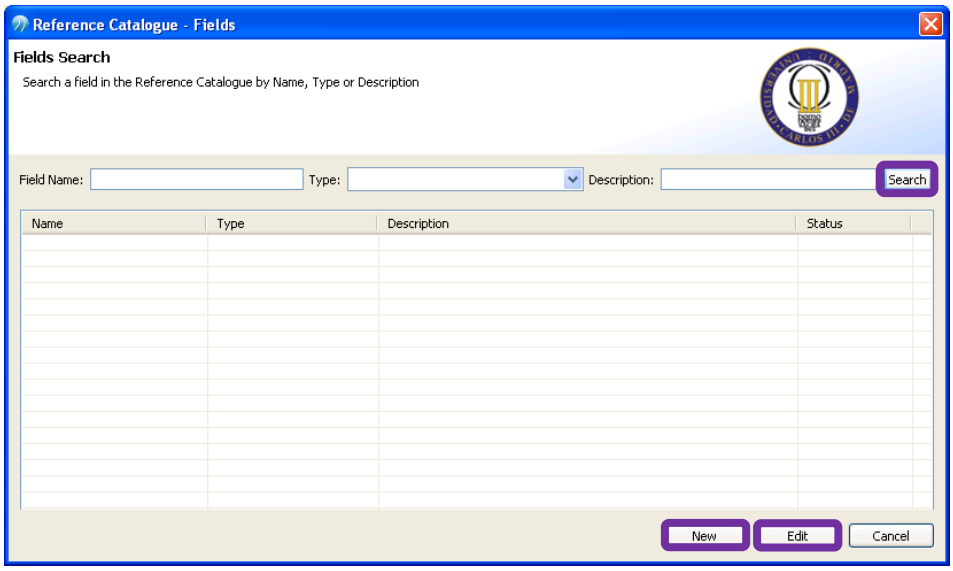


b. Presionar la opción del menú principal ***UC3M***. Una vez aparezcan todas las opciones, seleccionar:

Reference Catalogue → Fields



Nos aparecerá la siguiente ventana, ***Fields Search***, a partir de la cual podemos ***crear*** nuevos campos así como ***buscar*** y ***editar*** campos existentes. En los siguientes apartados, explicaremos el detalle de cada una de estas opciones.



C.1.1 Buscar Campo en el Catálogo de Referencia

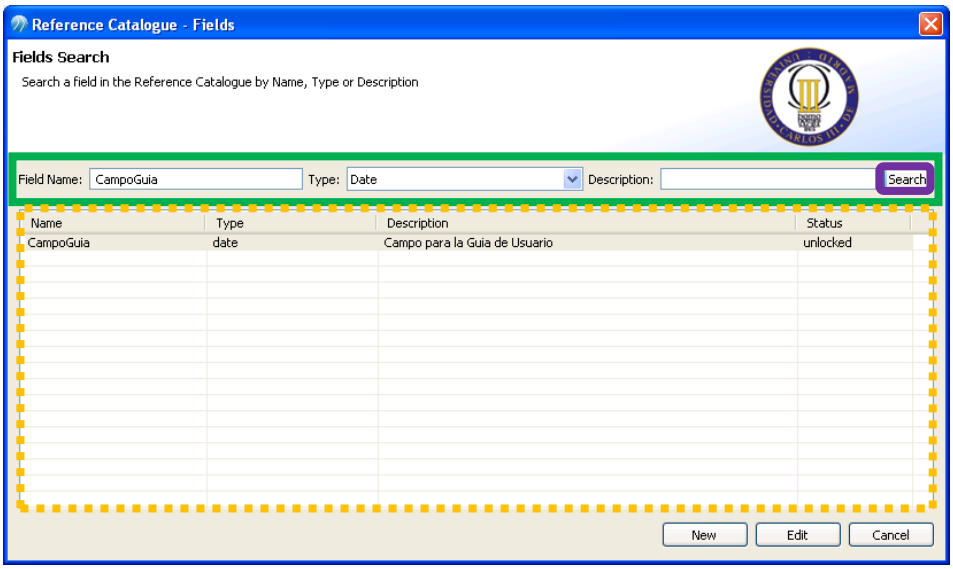
La herramienta *Field Generator* permite realizar búsquedas de artefactos *Field* en el catálogo, y por tanto comprobar la existencia de un campo en el Catálogo de Referencia:

1. Completar parcial o totalmente aquellos ***criterios*** por los que se desee realizar la búsqueda. En el caso de ejemplo, se ha añadido un valor de Field Name y el tipo del

campo (*Type*), pero también se podría usar como parámetro de búsqueda la descripción del campo (Description) así como cualquier combinación de estas variables.

2. Presionar botón ***Search***

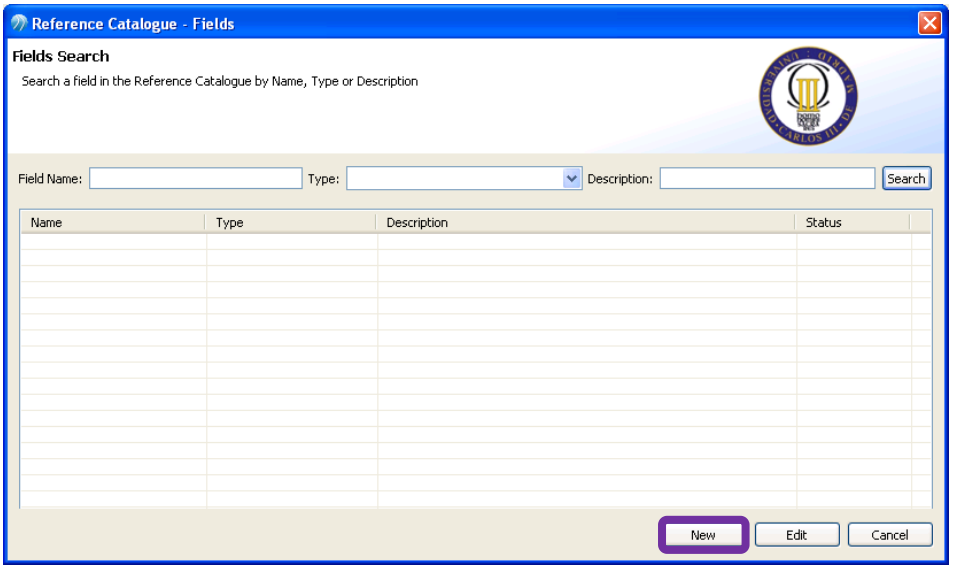
3. Aparecerá el ***listado de campos*** que cumplen las características indicadas en los parámetros de filtrado, además de su estado:
   - *Locked*, si el campo está siendo editado por otro usuario
   - *Unlocked*, si el campo está disponible para ser editado



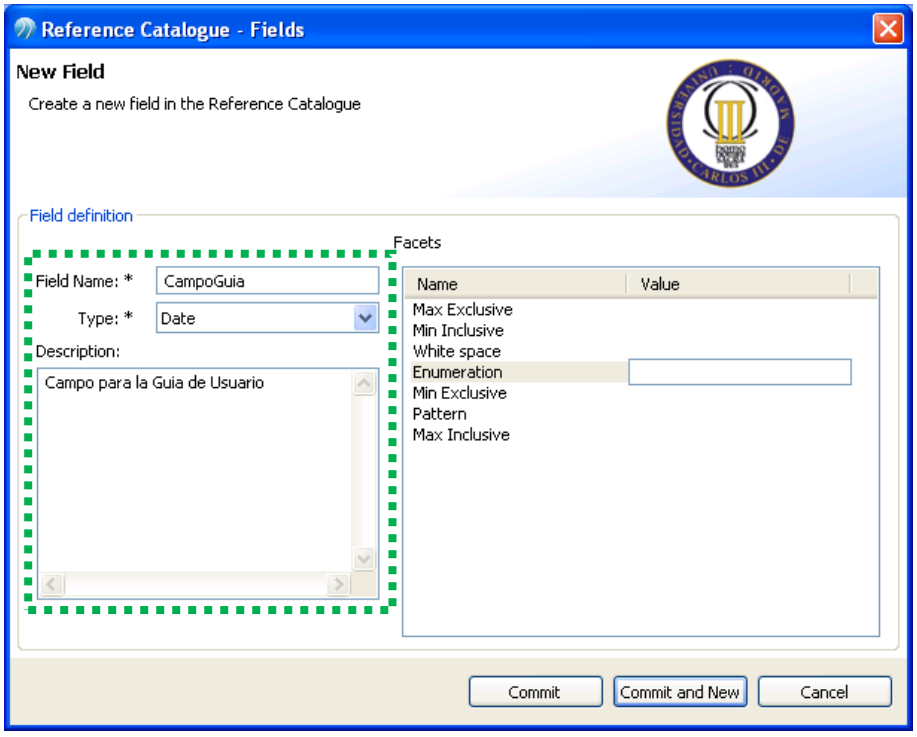
C.1.2 Crear Campo en el Catálogo de Referencia

Esta opción de la herramienta Field Generator permite crear un nuevo campo en el Catálogo de Referencia.

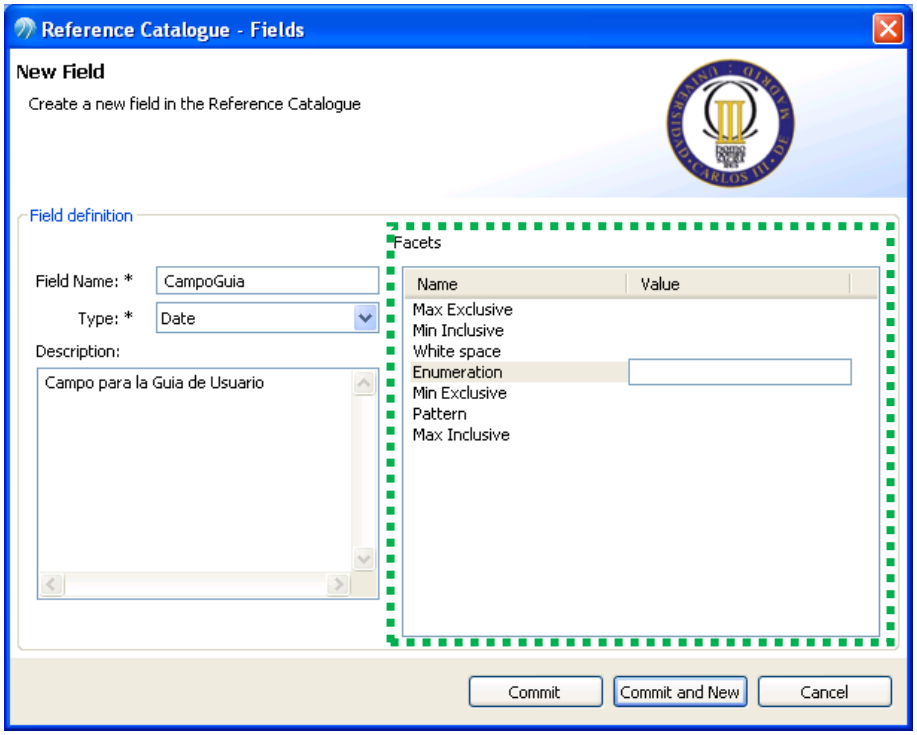
1. Presionar el botón ***New*** de la ventana principal de la herramienta *Field Generator.*



2. Completar el campo *Field Name* con el nombre del campo a incluir y seleccionar el tipo de entre las opciones que aparecen en el campo *Type*. Opcionalmente se puede incluir una descripción del campo en el área de texto *Description*.

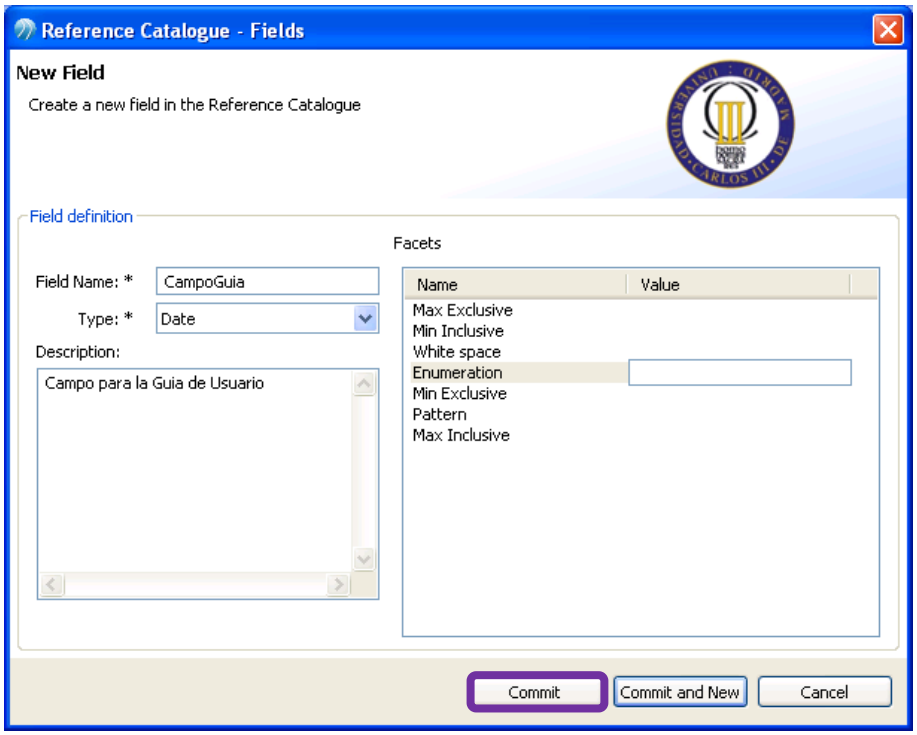


3. En función del tipo de campo seleccionado, aparecerán un conjunto de Facets que permiten acotar las características del campo. Se trata de parámetros opcionales para la creación del campo.



NOTA: Al pasar el ratón por encima del nombre de cada *facet* aparece la descripción.

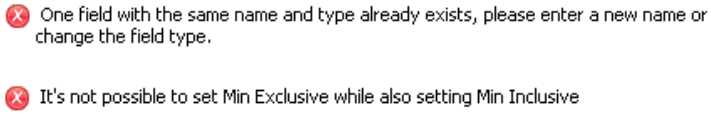
4. Presionar el botón ***Commit*** para registrar el campo en el Catálogo de Referencia.



NOTA: El botón ***Commit and New*** registra el campo y lanza una nueva ventana de creación de campo.

5. Para validar la creación del campo, se puede realizar la búsqueda del mismo tal y como se explica en el apartado Buscar Campo en el Catálogo de Referencia.

NOTA: Si el campo que se está intentando registrar existe previamente en el Catálogo (mismo *Field Name* y *Type* que otro campo existente) o las *Facets* cumplimentadas son incorrectas (por formato o incompatibilidades entre las mismas) aparecerá un mensaje en la cabecera de la ventana y no se registrará el artefacto en el Catálogo de Referencia. Ejemplos de mensajes de error:

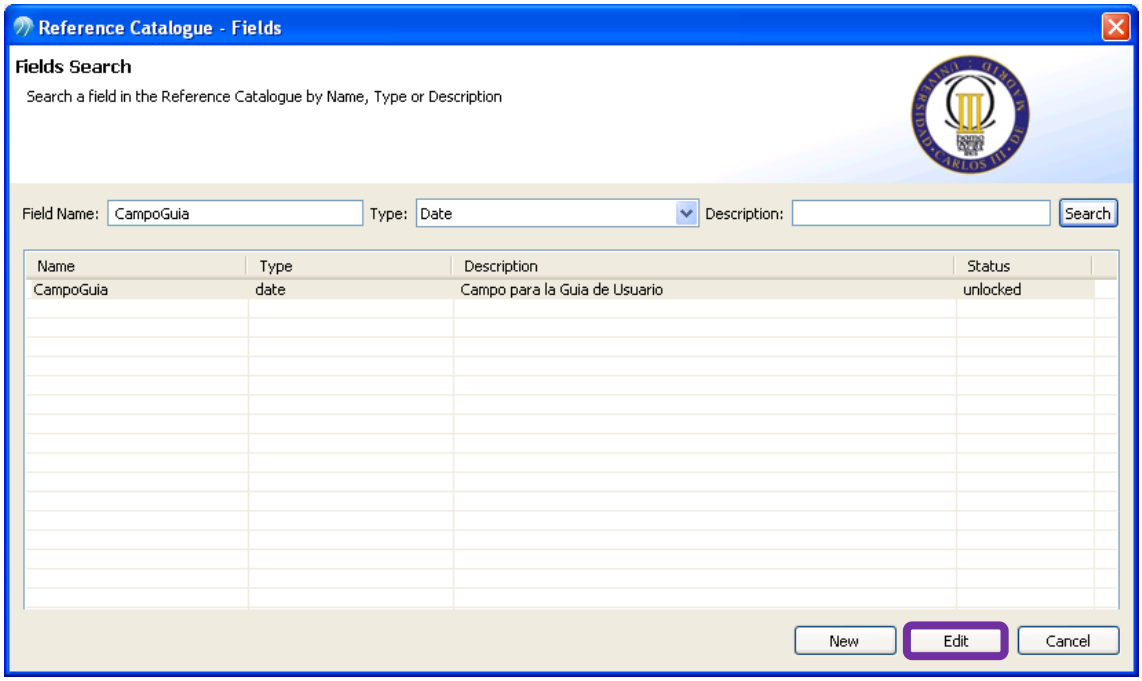


C.1.3 Editar Campo en el Catálogo de Referencia

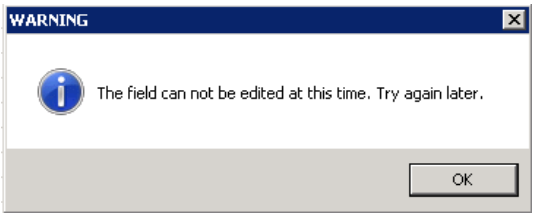
Esta opción de la herramienta Field Generator permite editar un campo en el Catálogo de Referencia.

1. Realizar la búsqueda del campo que se desea modificar mediante la opción búsqueda de la herramienta *Field Generator* detallada en el apartado Buscar Campo en el Catálogo de Referencia.

2. Seleccionar del listado el campo a editar y presionar el botón ***Edit*** (también doble click o intro).



NOTA: Los campos en estado *locked* no pueden ser abiertos en modo edición y se indicara mediante un popup al usuario.



3. Si el campo tiene estado *unlocked*, se abrirá la ventana con el detalle del campo, desde donde se podrá modificar la descripción y las restricciones (facets) del campo.

   NOTA: Al pasar el ratón por encima del nombre de cada facet aparece la descripción.

4. Una vez se han realizado las modificaciones, presionar el botón ***Commit*** para aplicar los cambios en el Catálogo.

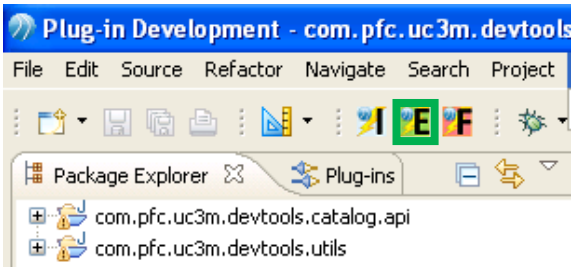
NOTA: Si las *Facets* cumplimentadas tienen errores de formato o son incompatibles entre ellas, aparecerá un mensaje en la cabecera de la ventana y no se actualizará el artefacto en el Catálogo de Referencia. Por ejemplo:

C.2 Entidades en el Catálogo de Referencia – Entity Generator

Esta ampliación de funcionalidad permite realizar el mantenimiento (búsqueda, creación, edición y exportación) de las Entidades del Catálogo de Referencia desde el IDE Eclipse del entorno local de desarrollo.

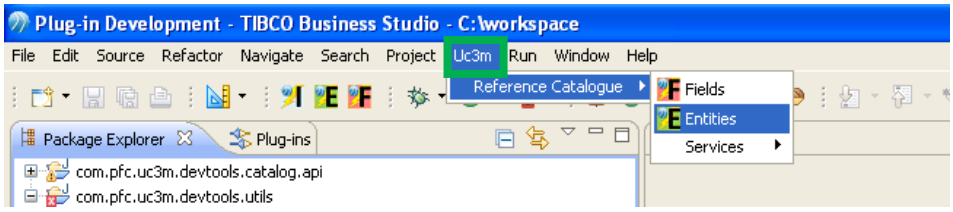
Para abrir la herramienta *Entity Generator*, que permite este mantenimiento de Entidades del Catálogo de Referencia, hay dos opciones:

a. Presionar el botón ***Entities*** de la barra de herramientas.

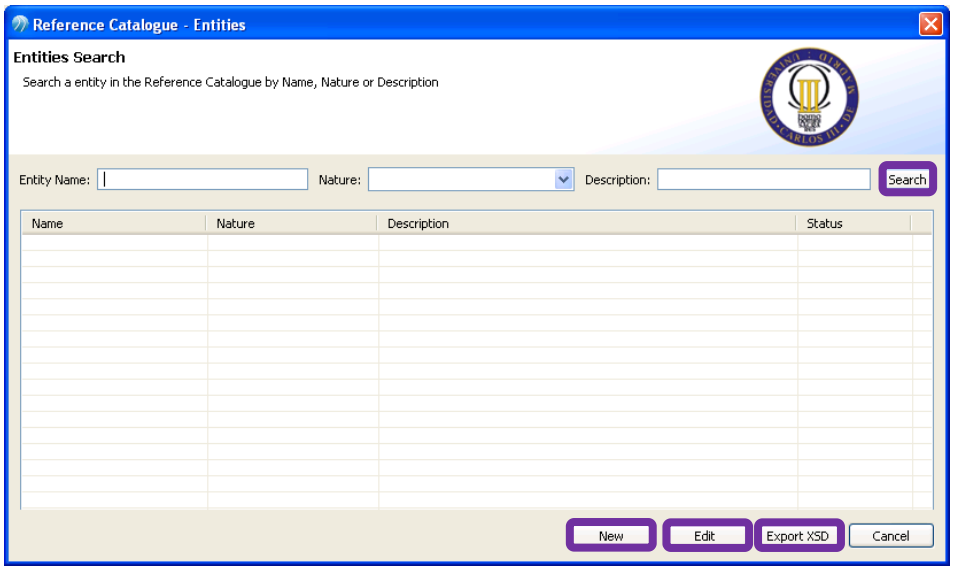


b. Presionar la opción del menú principal ***UC3M***. Una vez aparezcan todas las opciones, seleccionar:

Reference Catalogue → Entities



Nos aparecerá la siguiente ventana, ***Entities Search***, a partir de la cual podemos ***crear*** nuevas entidades así como ***buscar***, ***editar*** y ***exportar*** entidades ya existentes en el catálogo. En los siguientes apartados, explicaremos el detalle de cada una de estas opciones.



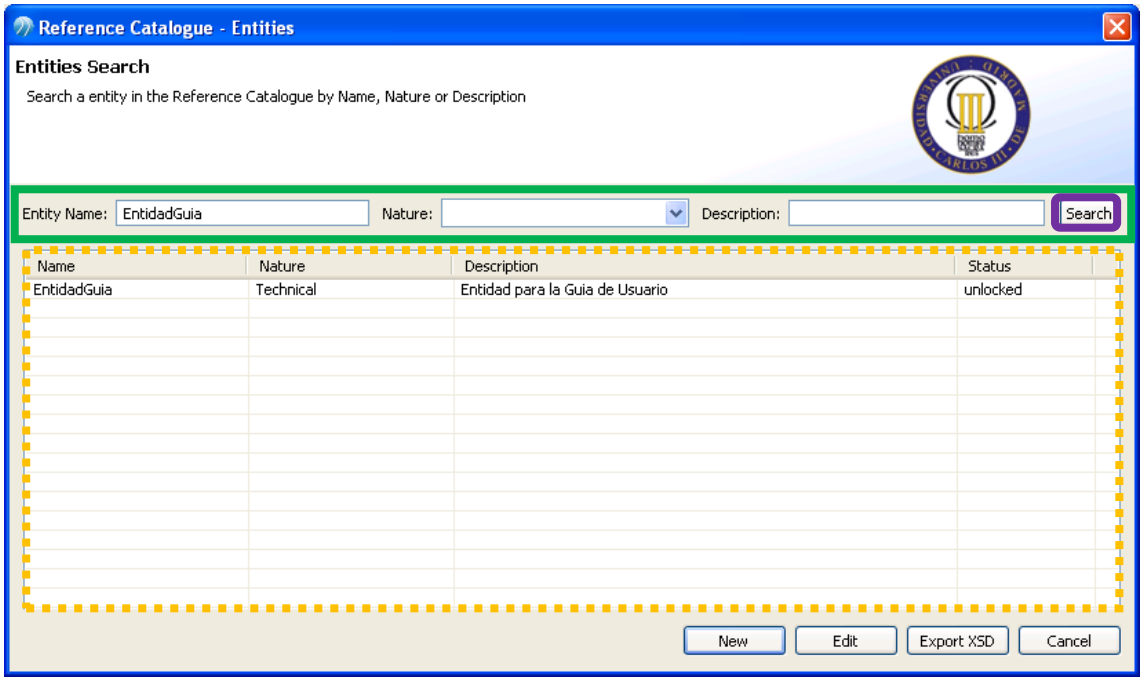
C.2.1 Buscar Entidad en el Catálogo de Referencia – Entities Search

Esta opción de la herramienta *Entity Generator* permite realizar búsquedas de artefactos *Entity* en el catálogo, y por tanto comprobar la existencia de una entidad en el Catálogo de Referencia:

1. Completar parcial o totalmente aquellos ***criterios*** por los que se desee realizar la búsqueda. En el caso de ejemplo, se ha realizado una búsqueda por nombre (*Name*), pero también se podría usar como parámetro de búsqueda la descripción (*Description*) o el tipo de la entidad (*Type*), así como cualquier combinación de estos parámetros.

2. Presionar botón ***Search***

3. Aparecerá el ***listado de entidades*** que cumplen las características indicadas en los parámetros de filtrado, además de su estado:
   - *Locked*, si la entidad está siendo editada por otro usuario
   - *Unlocked*, si la entidad está disponible para ser editada

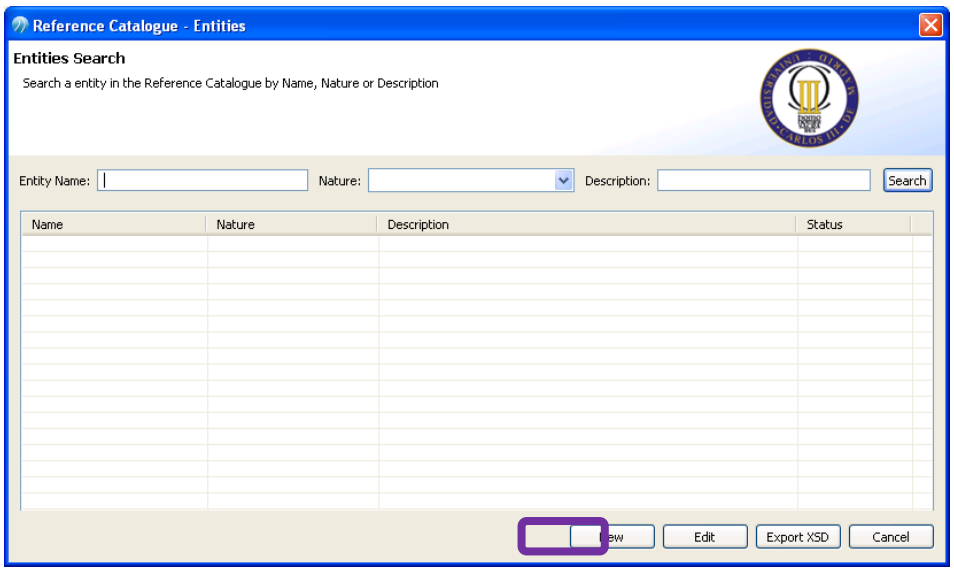


C.2.2 Crear Entidad en el Catálogo de Referencia

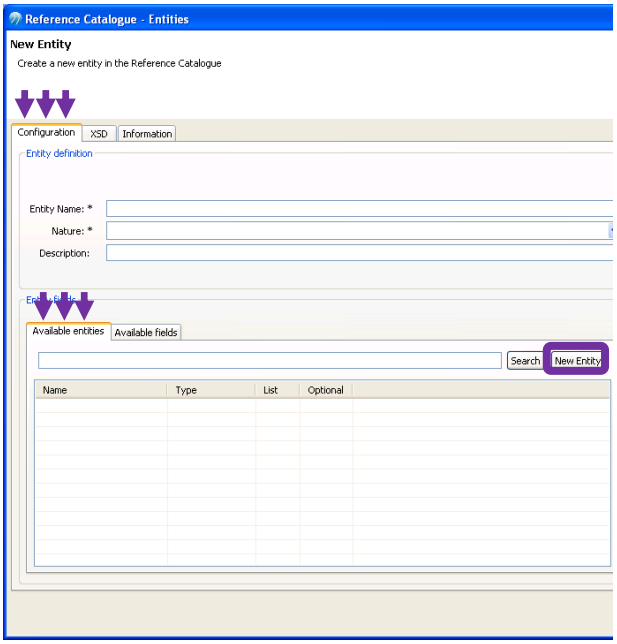
Esta opción de la herramienta *Entity Generator* permite registrar un nuevo elemento de tipo Entity en el Catálogo de Referencia.

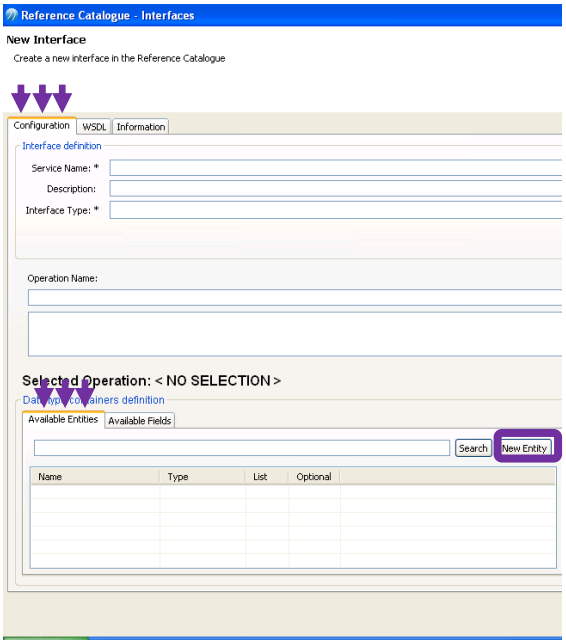
1. Acceder al editor de nuevas entidades (*Entity Editor*) a través de una de las siguientes opciones:

    a) Presionar el botón ***New*** de la ventana principal de la herramienta *Entity Generator.*



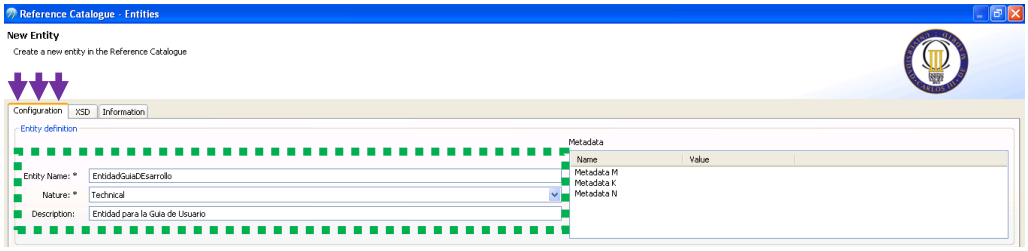
    b) Presionar el botón ***New Entity*** desde el área de selección de entidades (*Available Entities*) de un editor de entidades o interfaces.



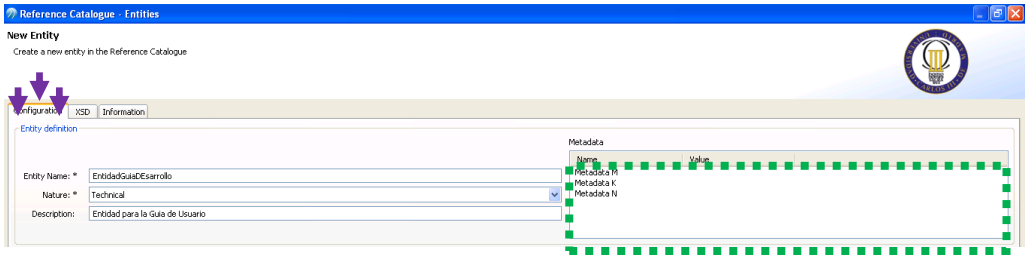


2. Cumplimentar el área de definición de la entidad, ***Entity Definition***.

    a) Completar el campo *Entity Name* con el nombre de la entidad a incluir y seleccionar el tipo entre las opciones que aparecen en la combo *Type* (Business / Technical). Opcionalmente se puede incluir una descripción de la entidad en el área de texto *Description*.



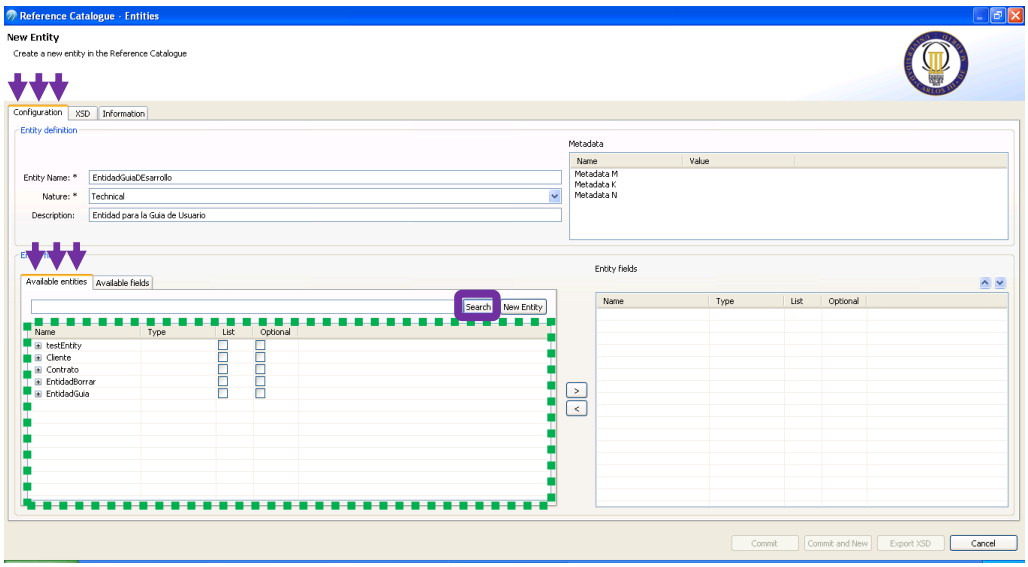
b) Adicionalmente, se podrá agregar información funcional a través de la tabla de ***metadatos***. Estos metadatos deberán haber sido definidos previamente por los equipos funcionales en un *properties* de la propia herramienta de generación.



3. A continuación, definir los elementos que componen la entidad. Estos elementos pueden ser campos u otras entidades definidas previamente y que existan en el Catálogo de Referencia.

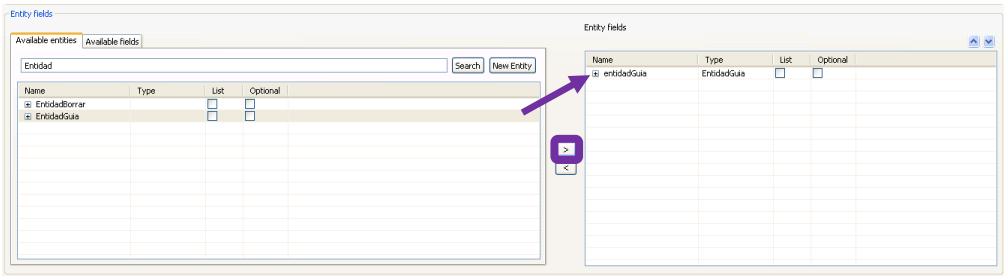
   a) Inclusión de elementos a partir de entidades ya existentes.

   a.1) Realizar la búsqueda de la Entidad a incluir desde la sección de *Entity Fields* de la ventana: incluir el nombre de la entidad a filtrar en la caja de texto de la pestaña *Available entities* y presionar el botón ***Search***.

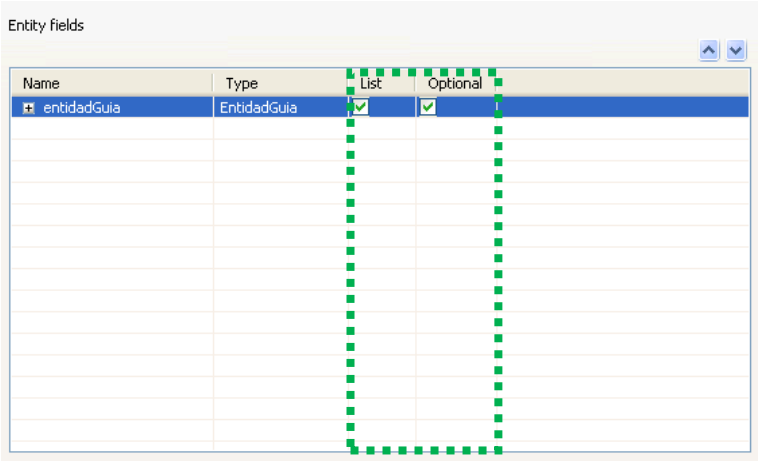


NOTA: Las entidades resultado de la búsqueda pueden ser expandidas (⊞) para visualizar sus componentes.

a.2) Seleccionar de la tabla de resultados, la entidad o componente de la misma que se desee incluir; a continuación presionar el botón "**>**" para agregar el elemento a la definición de la nueva entidad.



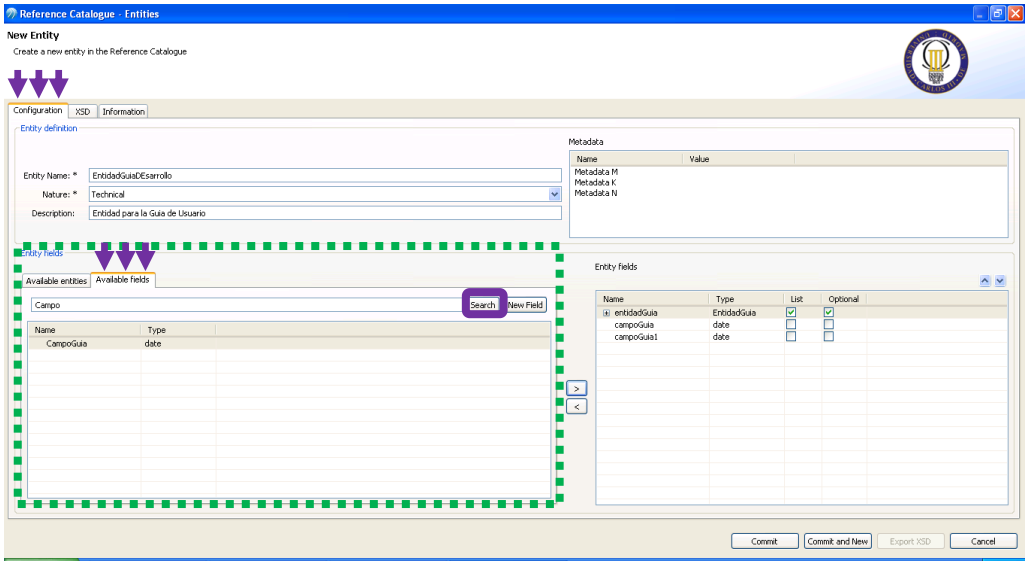
a.3) Una vez el elemento seleccionado aparezca en el listado de campos de la entidad, podemos definir las ocurrencias del mismo: si el componente aparecerá como un listado (check sobre checkbox *List*) o si será opcional (check sobre checkbox *Optional*) en la nueva entidad.



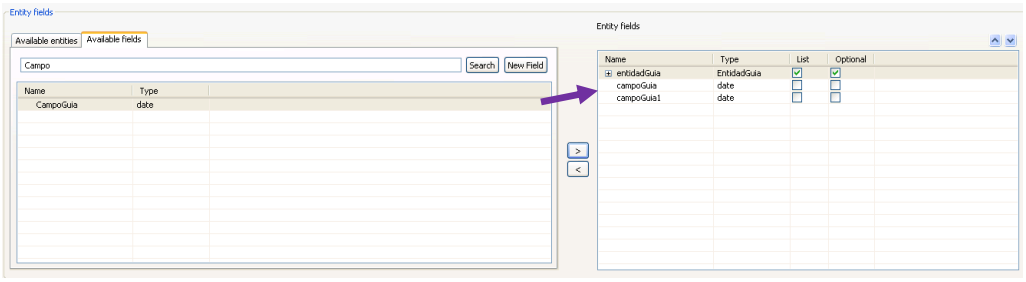
a.4) Repetir el procedimiento por cada una de los elementos procedentes de entidades que se quieran incluir.

b) Inclusión de ***campos*** como elemento de la nueva Entity.

b.1) Realizar la búsqueda del campo a incluir desde la sección de *Available fields* de la ventana para crear una nueva Entidad del Entity Generator. Para ello, incluir el nombre del campo a filtrar en el campo *Available fields* y presionar el botón *Search*.

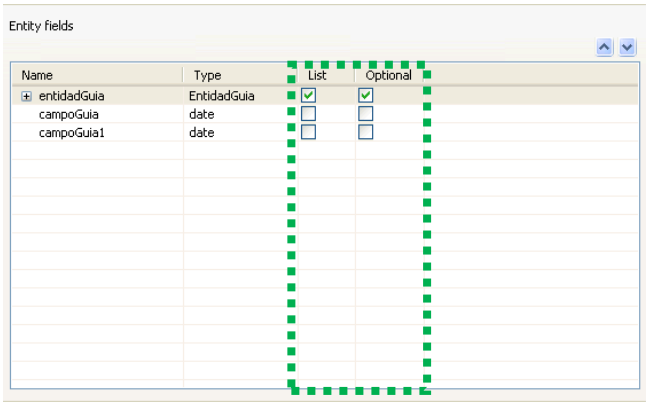


b.2) Seleccionar de la tabla de resultados, el campo o campos que se deseen; a continuación presionar el botón “>” para agregarlos como elementos de la nueva Entidad que estamos definiendo.

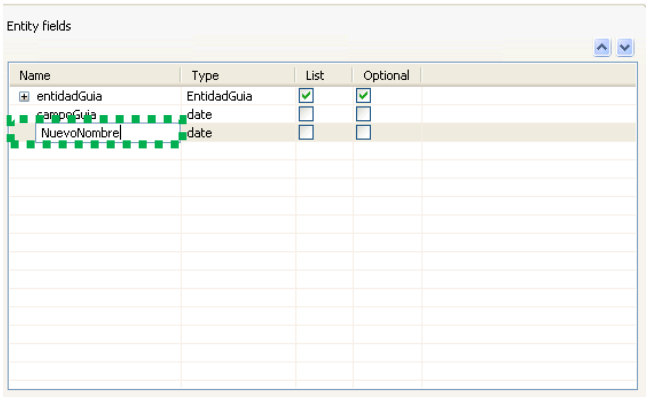


NOTA: Un componente puede ser agregado varias veces a la misma entidad.

b.3) Una vez el campo seleccionado aparezca en el listado de campos de la entidad, podemos definir las ocurrencias del mismo: si el campo aparecerá como un listado en la nueva entidad (check sobre checkbox *List*) o si será opcional (check sobre checkbox *Optional*).

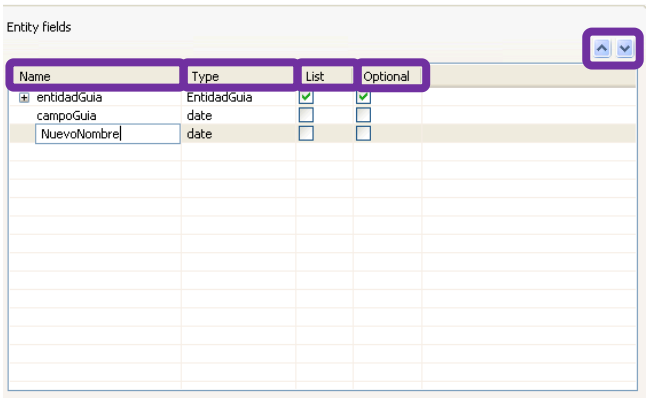


b.4) Si se desea se pueden renombrar los elementos que componen la entidad, de modo que su nomenclatura sea más intuitiva.

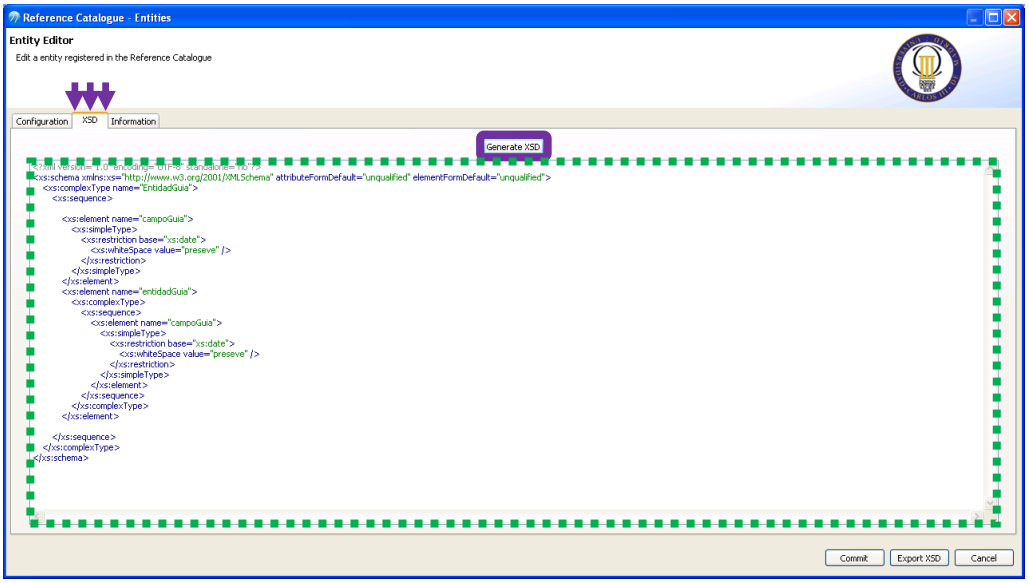


b.5) Repetir el procedimiento para cada uno de los campos que se quieran incluir en la definición de la nueva entidad.

4. Ordenar los campos según el orden deseado: se pueden realizar ordenaciones personalizadas o alfabéticas a través de las cabeceras.

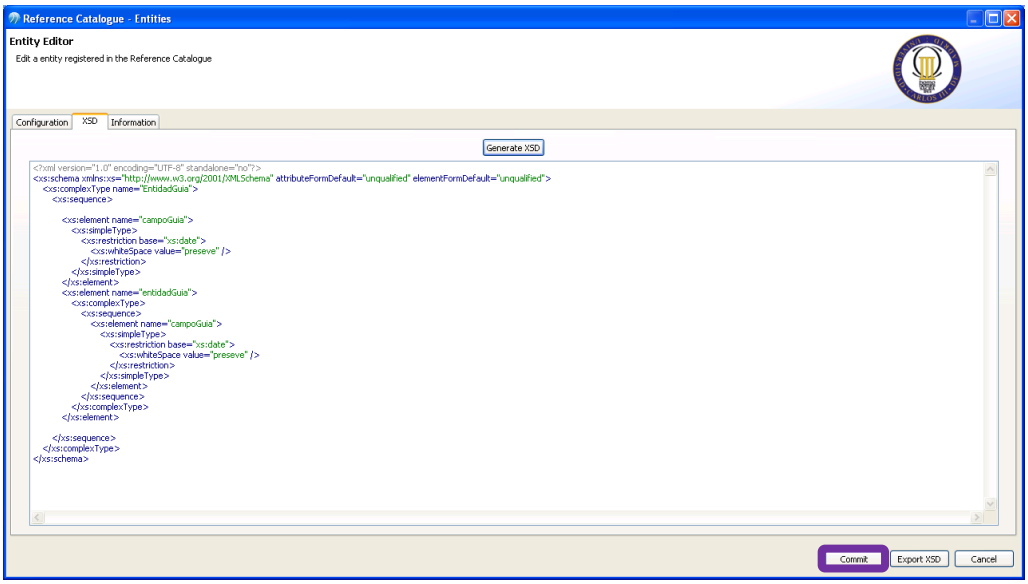


5. Presionar el botón de ***Generate XSD*** para comprobar que la entidad creada tiene la ***estructura*** esperada.

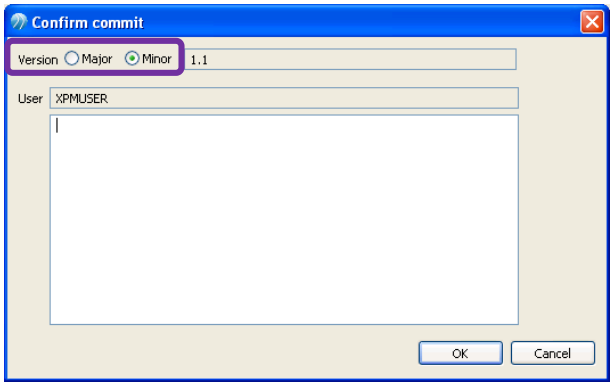


6. Registrar la entidad en el Catálogo de Referencia

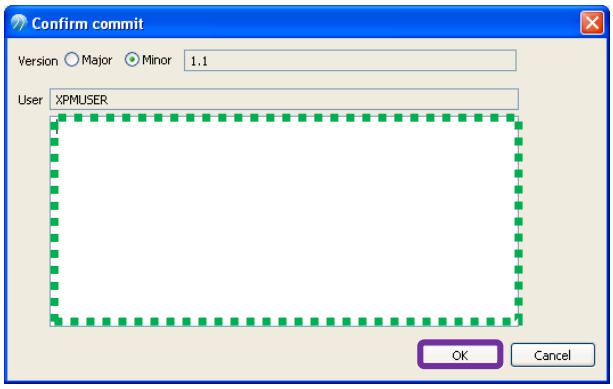
    a) Presionar el botón ***Commit***.



b) Seleccionar si será una versión ***Minor*** o ***Major***.



c) Añadir un ***comentario*** de versión si se desea y aceptar el registro.



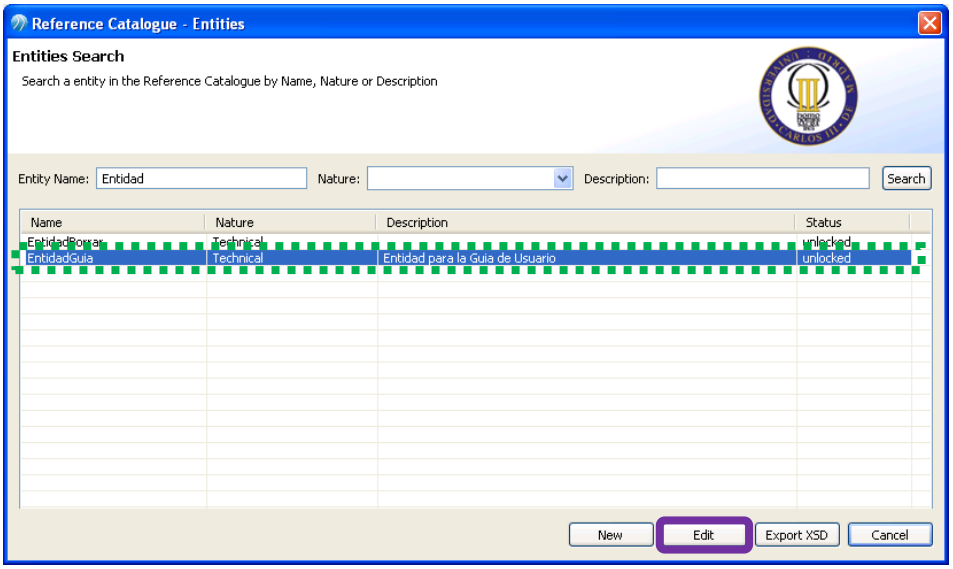
7. Para validar la creación de la Entidad, realizar la búsqueda de la Entidad tal y como se ha explicado en el apartado Buscar Entidad en el Catálogo de Referencia.

C.2.3 Editar Entidad en el Catálogo de Referencia

Esta opción de la herramienta *Entity Generator* permite editar una entidad en el Catálogo de Referencia.

1. Realizar la búsqueda de la entidad que se desea modificar mediante la opción búsqueda de la herramienta *Entity Generator* detallada en el apartado Buscar Entidad en el Catálogo de Referencia.

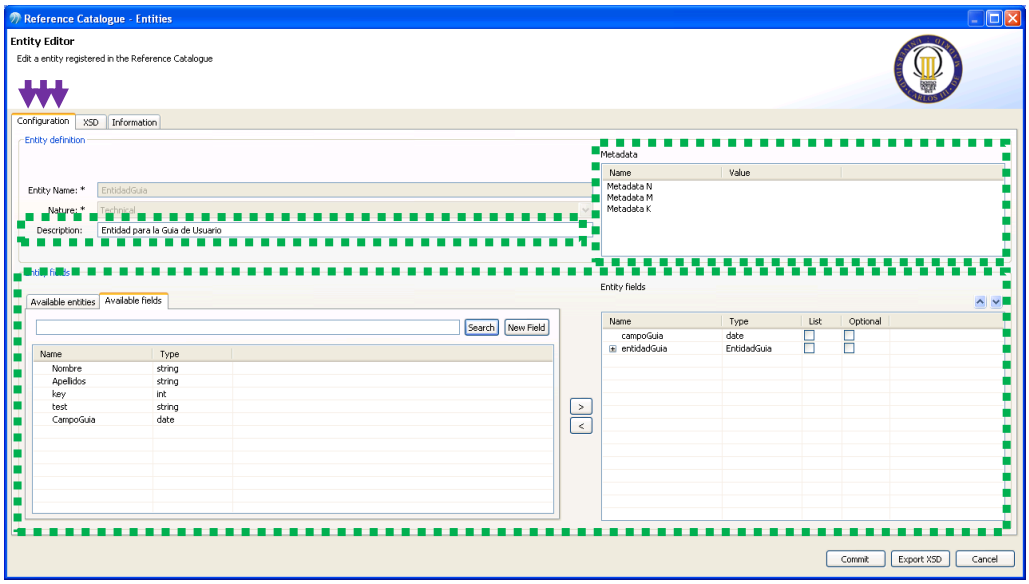
2. Seleccionar del listado la entidad a editar y presionar el botón ***Edit*** (también doble click o intro).



3. Editar la entidad teniendo en cuenta que los campos modificables son:

    a) En el área *Entity definition*: texto *Description* y valores de la tabla *Metadata*.

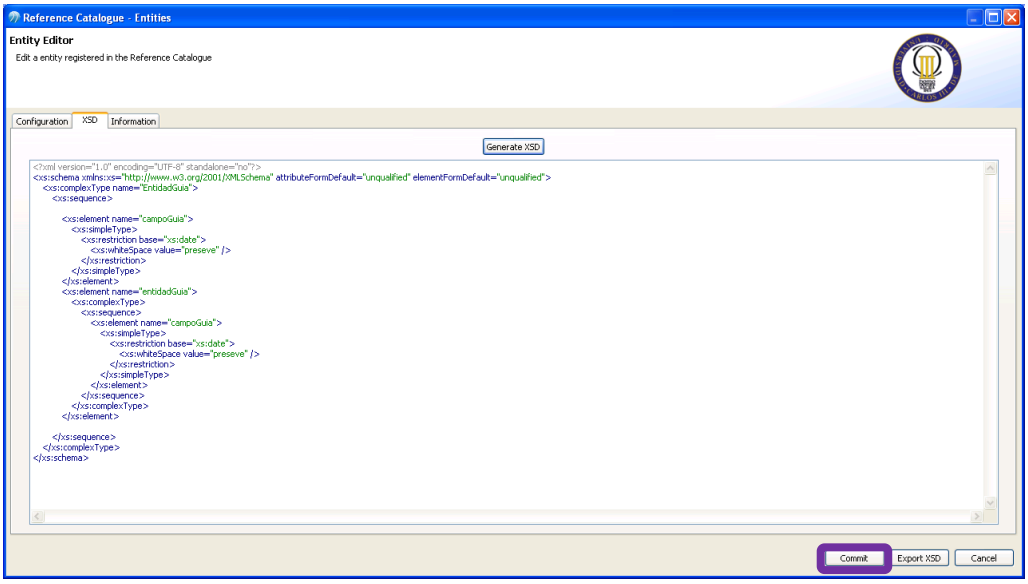
    b) En el área Entity fileds: elementos a la entidad (añadir/quitar, nomenclatura y ocurrencias).



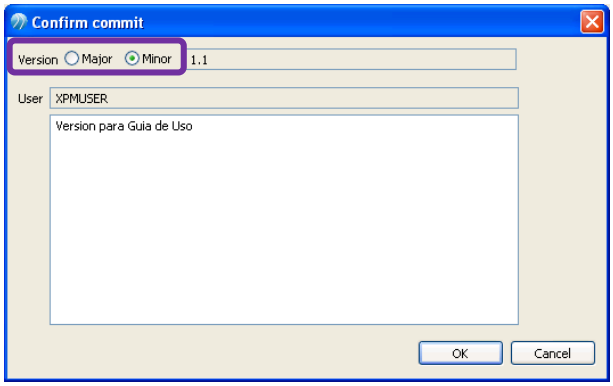
4. Presionar el botón ***Generate XSD*** para comprobar que las modificaciones en la entidad son las deseadas.

5. Finalmente, aplicar los cambios de la entidad sobre Catálogo de Referencia

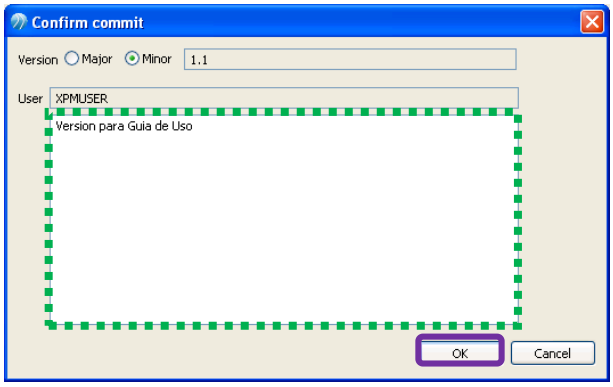
    a) Presionar el botón ***Commit***.



    b) Seleccionar si será una versión ***Minor*** o ***Major***.



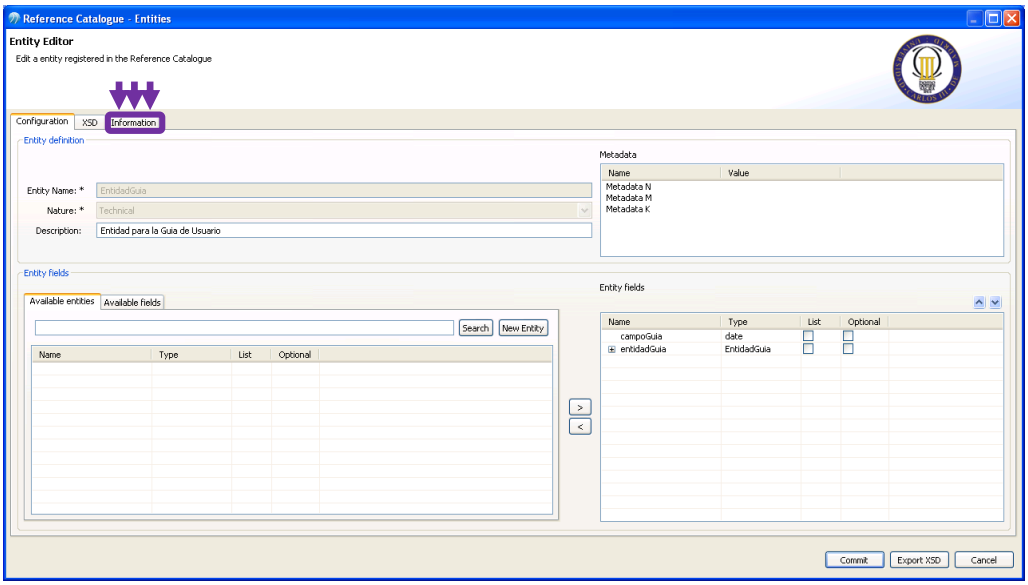
c) Añadir un ***comentario*** de versión si se desea y aceptar el registro.



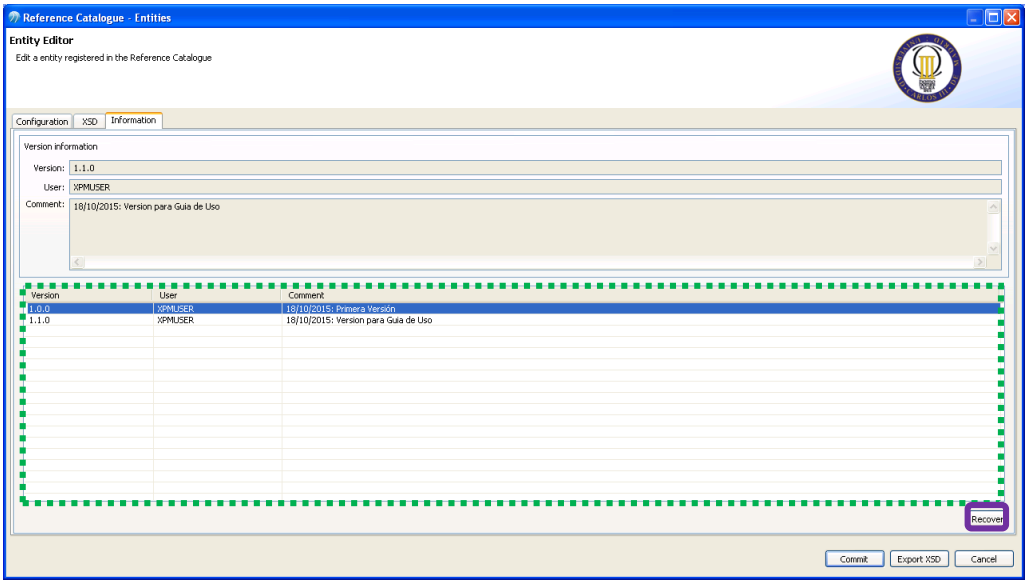
C.2.4 Recuperar Entidad del Catálogo de Referencia

Esta opción de la herramienta *Entity Generator* permite recuperar versiones anteriores de una Entidad guardadas en el Catálogo de Referencia. Para recuperar versiones anteriores, sólo puede hacerse editando una Entidad ya creada.

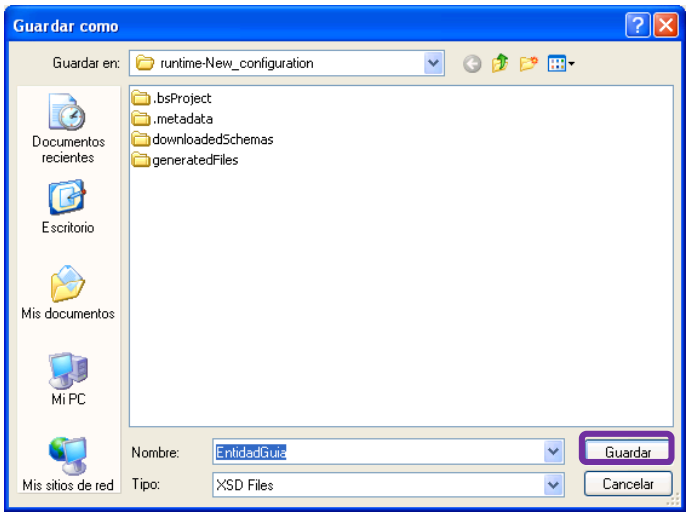
1. Una vez seleccionada una Entidad para editar y presionar *Edit*, presionar la pestaña ***Information***.



2. Seleccionar la versión que se quiere recuperar de la tabla y presionar ***Recover***.



3. Finalmente, navegar por el sistema de ficheros para seleccionar la ruta donde se desea descargar la versión anterior de la Entidad y presionar ***Save (Guardar)***.



C.2.5 Exportar Entidad del Catálogo de Referencia

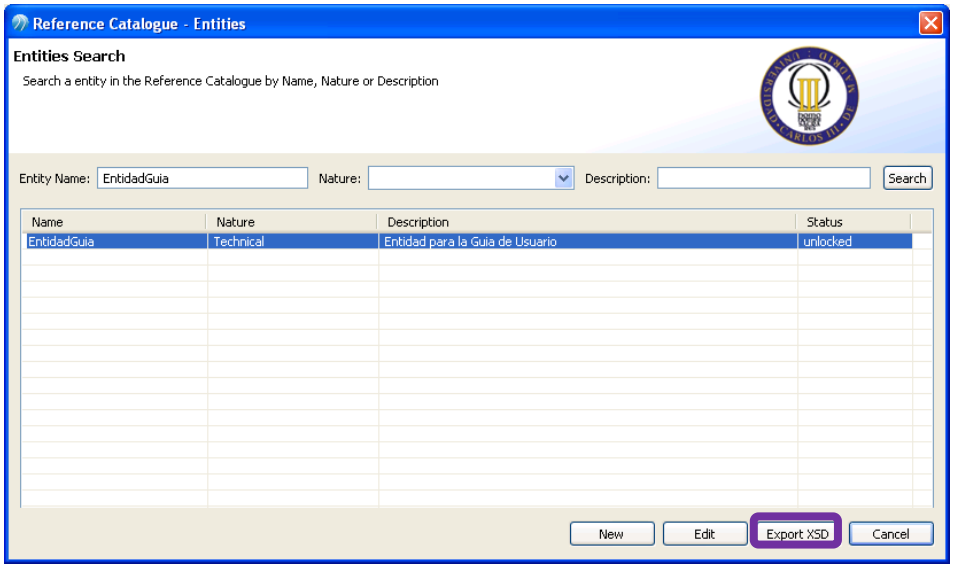
Esta opción de la herramienta *Entity Generator* permite exportar el schema xsd asociado a un elemento de tipo Entity albergado en el Catálogo de Referencia.

1. Se puede invocar a la función de exportación de entidades desde:

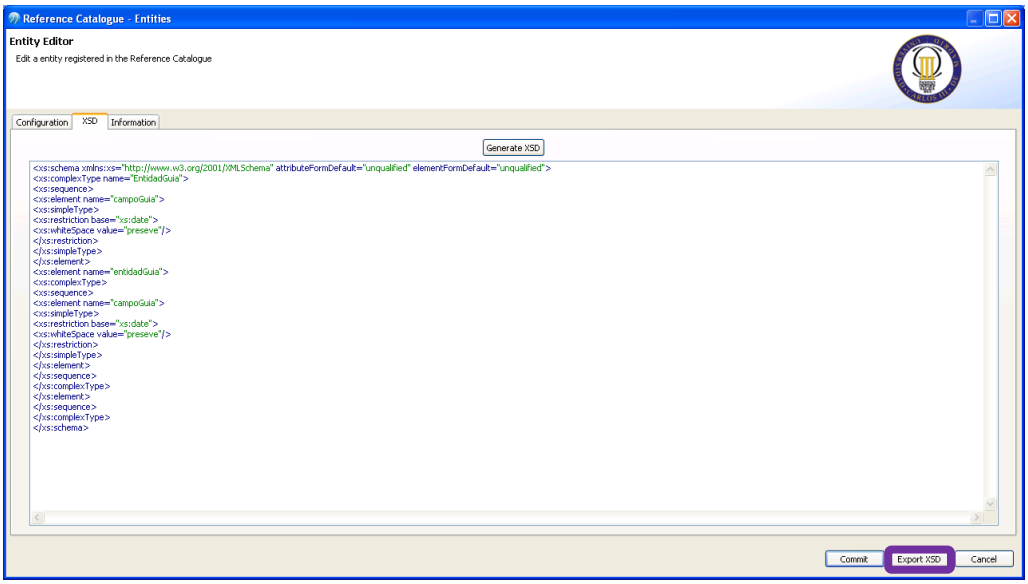
    a) Ventana principal de la herramienta *Entity Generator:*

        a.1) Realizar la búsqueda de la entidad que se desea exportar mediante la opción búsqueda de la herramienta *Entity Generator* detallada en el apartado Buscar Entidad en el Catálogo de Referencia.

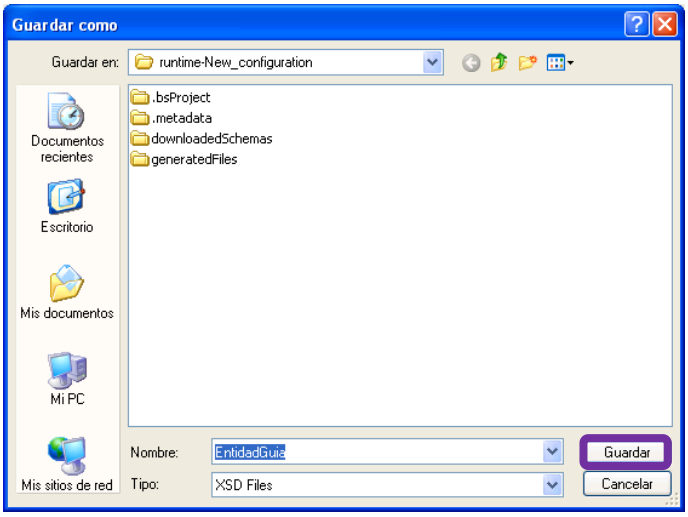
a.2) Seleccionar del listado de resultados la entidad a exportar y presionar el botón ***Export XSD***.



b) Ventana de edición de entidades, *Entity Editor:*



2. Finalmente, aparecerá una ventana de salvado (Save As): una vez seleccionada la ruta y nombre de salvado en local (por defecto, la ruta destino es la del workspace de trabajo), presionar ***Save (Guardar)*** para completar la exportación.

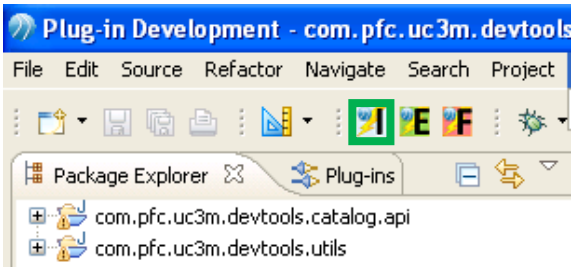


C.3 Interfaces en el Catálogo de Referencia – Interface Generator

Esta ampliación de funcionalidad permite realizar el mantenimiento (búsqueda, creación, edición y exportación) de las Interfaces del Catálogo de Referencia desde el Eclipse del entorno local de desarrollo.

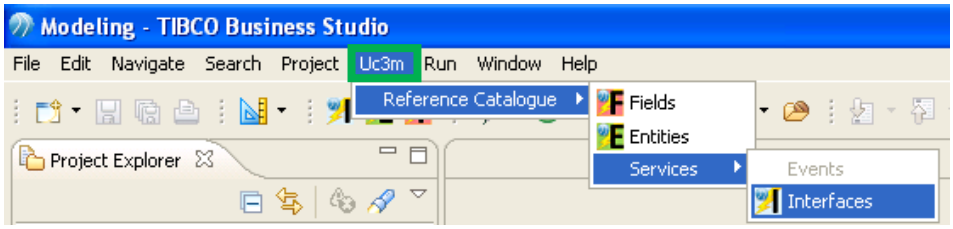
Para abrir la herramienta *Interface Generator*, que permite este mantenimiento de interfaces del Catálogo de Referencia, hay dos opciones:

a. Presionar el botón ***Interfaces*** de la barra de herramientas.

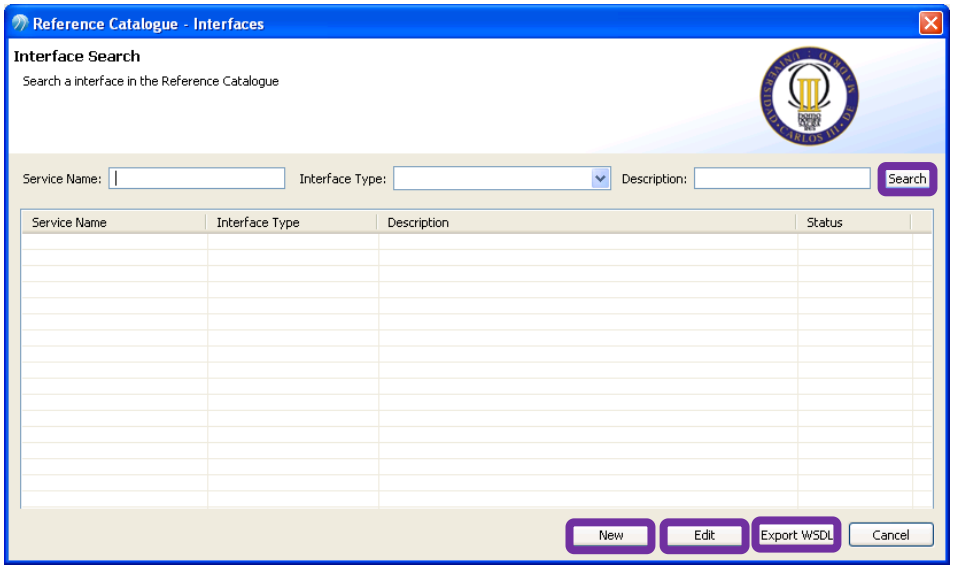


b. Presionar la opción del menú principal ***UC3M***. Una vez aparezcan todas las opciones, seleccionar:

Reference Catalogue → Services → Interfaces



Nos aparecerá la siguiente ventana, ***Interfaces Search***, a partir de la cual podemos ***crear*** nuevas interfaces así como ***buscar***, ***editar*** y ***exportar*** interfaces ya existentes en el catálogo. En los siguientes apartados, explicaremos el detalle de cada una de estas opciones.



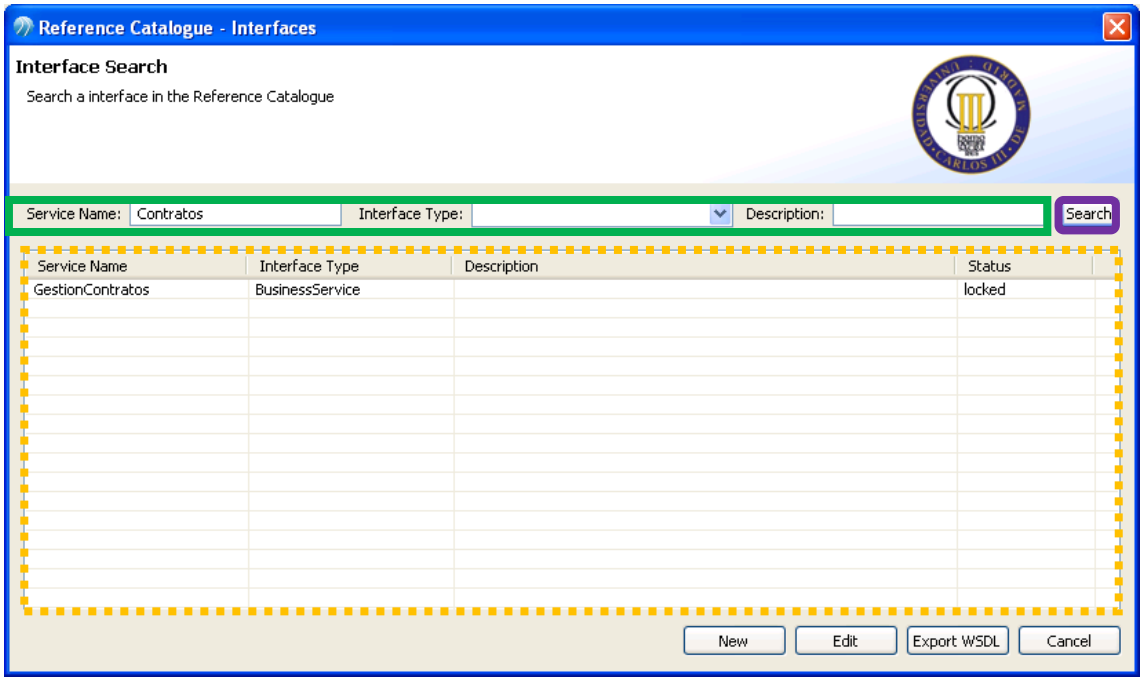
C.3.1 Buscar Interfaz en el Catálogo de Referencia – Interfaces Search

Esta opción de la herramienta *Interface Generator* permite realizar búsquedas de artefactos *Service* en el catálogo, y por tanto comprobar la existencia de una interface en el Catálogo de Referencia:

1. Completar parcial o totalmente aquellos ***criterios*** por los que se desee realizar la búsqueda. En el caso de ejemplo, se ha realizado una búsqueda por nombre (Service *Name*) y tipo (Interface Type), pero también se podría usar como parámetro de búsqueda la descripción (*Description*), así como cualquier combinación de estos parámetros.

2. Presionar botón ***Search***

3. Aparecerá el ***listado de servicios*** que cumplen las características indicadas en los parámetros de filtrado, además de su estado:
   - *Locked*, si la interface del servicio está siendo editada por otro usuario
   - *Unlocked*, si la interface del servicio está disponible para ser editada

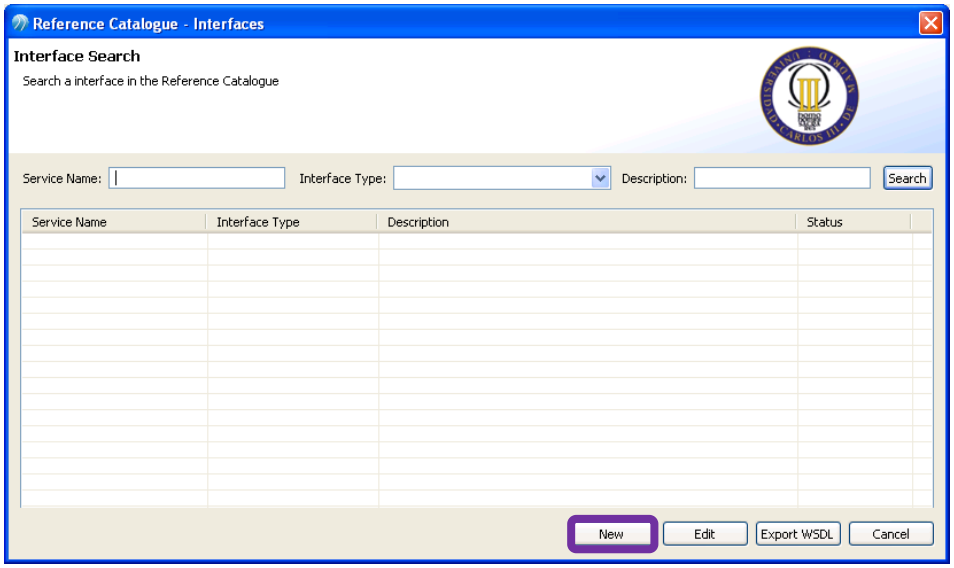


C.3.2 Crear Interfaz en el Catálogo de Referencia

Esta opción de la herramienta *Interface Generator* permite registrar un nuevo elemento de tipo *Service* en el Catálogo de Referencia.

1. Acceder al editor de nuevas interfaces (*Interface Editor*) a través de una de las siguientes opciones:

    a) Presionar el botón ***New*** de la ventana principal de la herramienta *Interface Generator.*

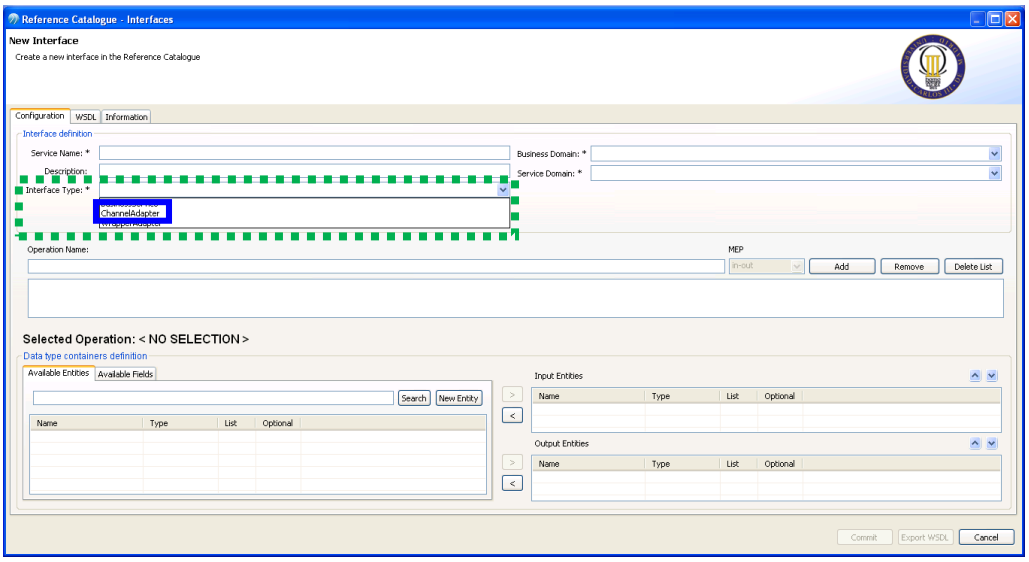


2. Se abrirá una nueva ventana, para detallar la información de la Interface a crear. En función del tipo de Interface a crear, la información a incluir en el área *Interface definition* será distinta. A continuación se expone cuáles son las diferencias para cada tipo de servicio.

C.3.2.1 Interface definition para Interfaz de tipo Channel Adapter

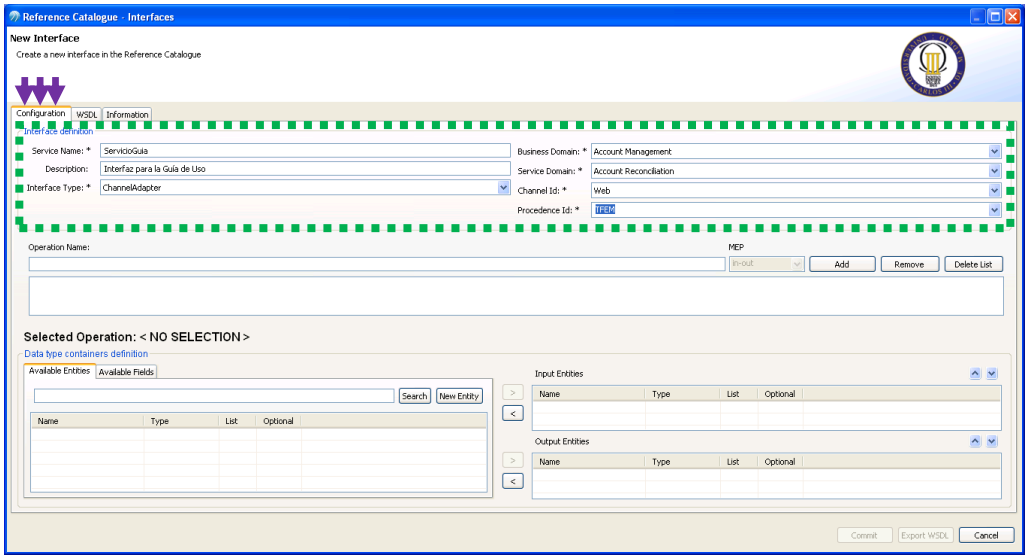
En este apartado se detalla qué información se debe incluir la *Interface definition* para un Channel Adapter.

1. Seleccionar del combo ***Interface Type*** el tipo **Channel Adapter**

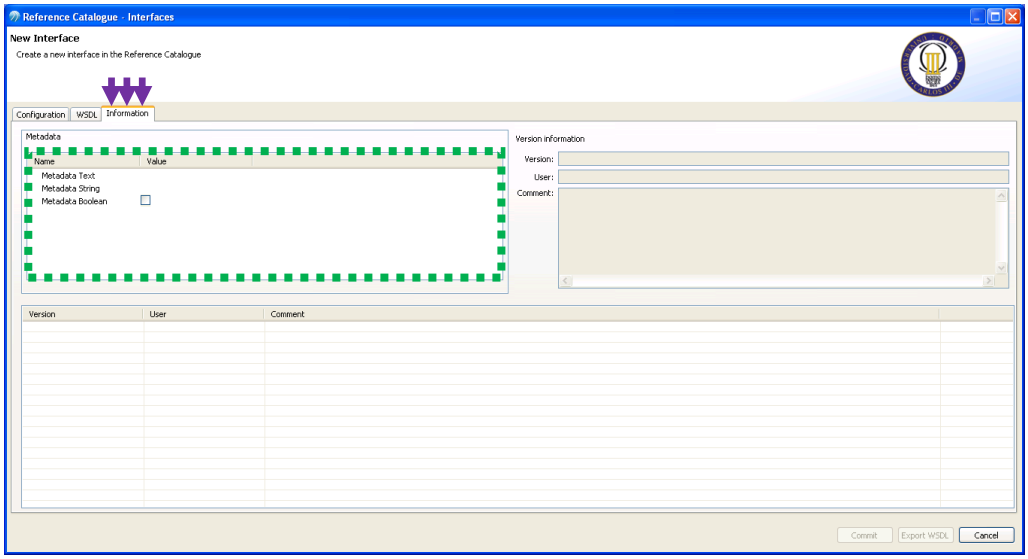


2. Completar los campos que componen la definición de la Interfaz del Channel Adapter:

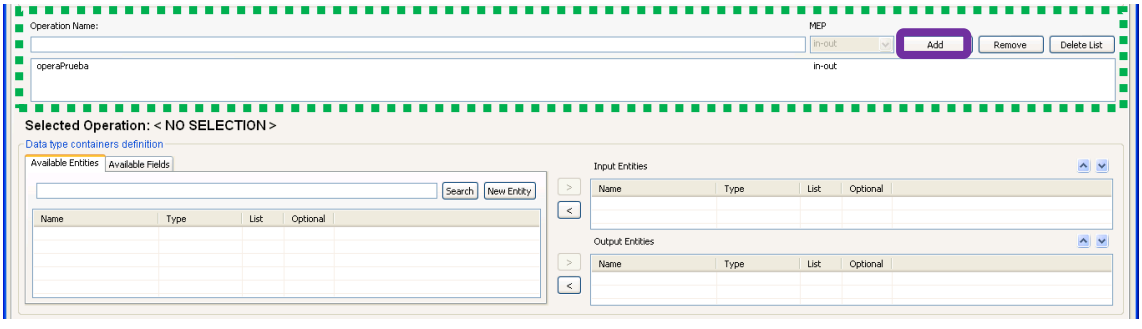
    a) ***Interface Definition***: *Service Name* (el nombre del servicio a crear), *Description* (definición de la interfaz a crear), *Business Domain* (Business Domain en el que se incluye el servicio), *Service Domain* (Service Domain en el que se incluye el servicio), *Channel Id* (identificador del tipo de canal: WEB/MOB) y *Procedence Id* (identificador de la procedencia del canal).



b) ***Metadata***: Valores funcionales para los campo definidos en la tabla *Metadata* de la pestaña *Information.*



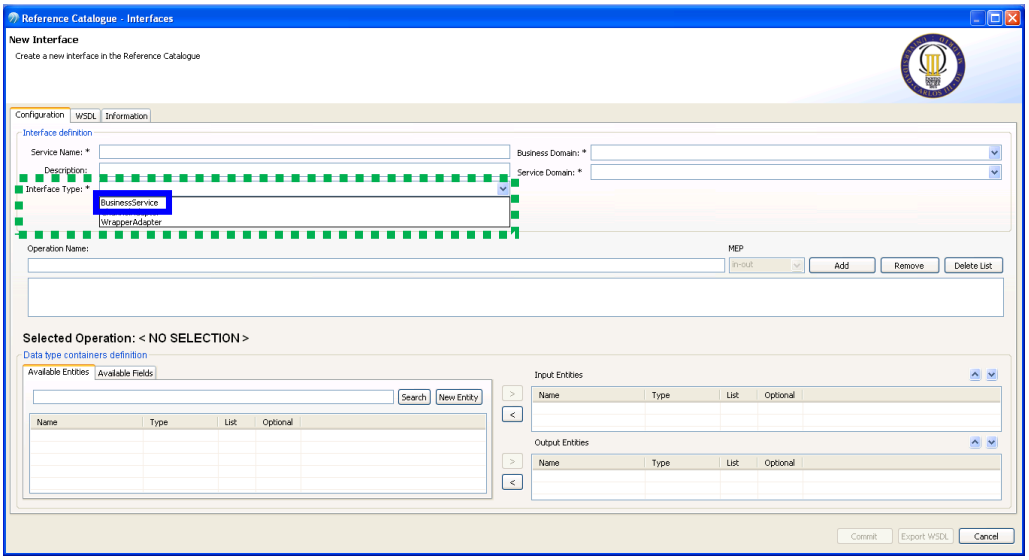
3. Añadir una operación a la definición de la Interfaz. Para ello, escribir su nombre en ***Operation Name*** y presionar el botón ***Add***. La operación aparecerá en el listado de operaciones incluidas en la Interfaz.



C.3.2.2 Interface definition para Interfaz de tipo Business Service

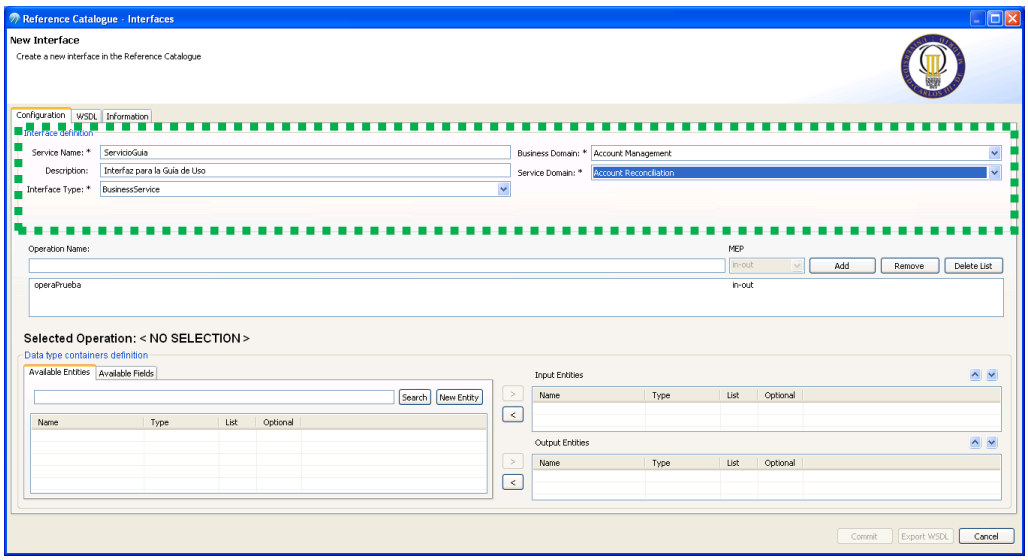
En este apartado se detalla qué información se debe incluir la *Interface definition* para un Business Service.

1. Seleccionar del combo ***Interface Type*** el tipo **Business Service**

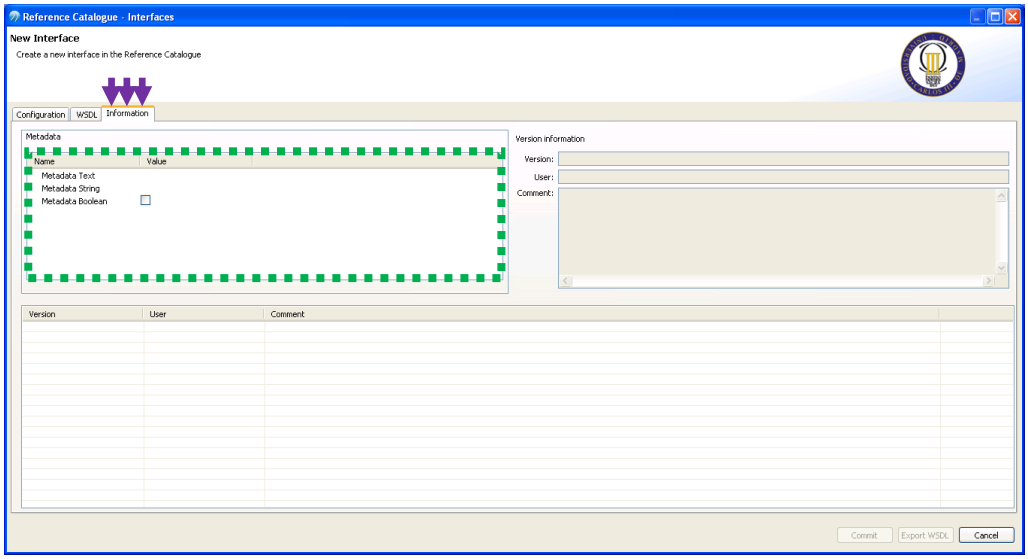


2. Completar los campos que componen la definición de la Interfaz del Busimess Service:

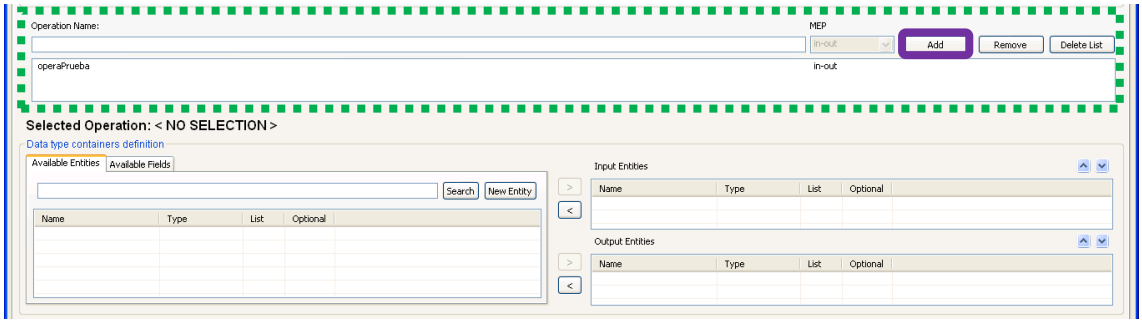
   a) ***Interface Definition***: *Service Name* (el nombre del servicio a crear), *Description* (definición de la interfaz a crear), *Business Domain* (Business Domain en el que se incluye el servicio) y *Service Domain* (Service Domain en el que se incluye el servicio).



   b) ***Metadata***: Valores funcionales para los campo definidos en la tabla *Metadata* de la pestaña *Information.*



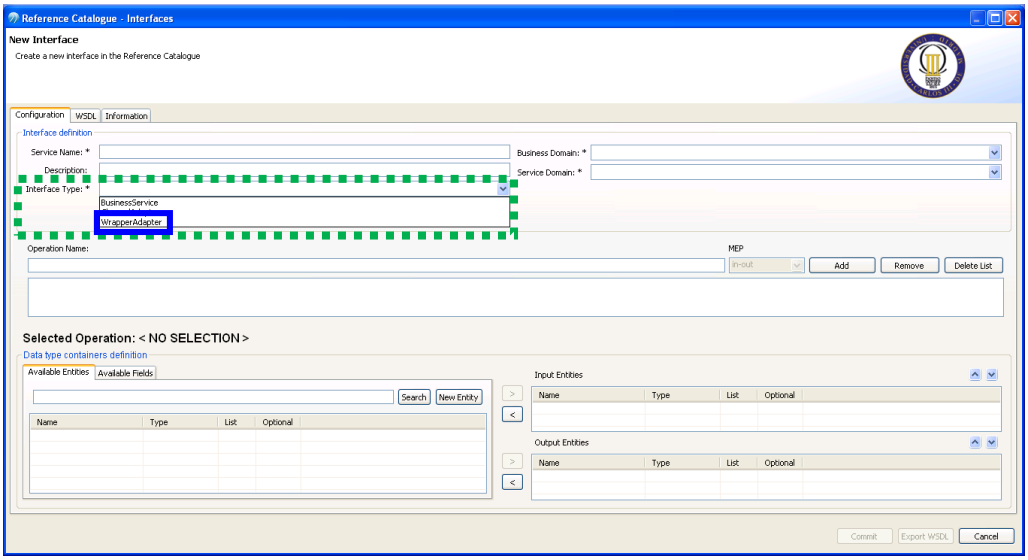
3. Añadir una operación a la definición de la Interfaz. Para ello, escribir su nombre en ***Operation Name*** y presionar el botón ***Add***. La operación aparecerá en el listado de operaciones incluidas en la Interfaz.



C.3.2.3 Interface definition para Interfaz de tipo Wrapper Adapter

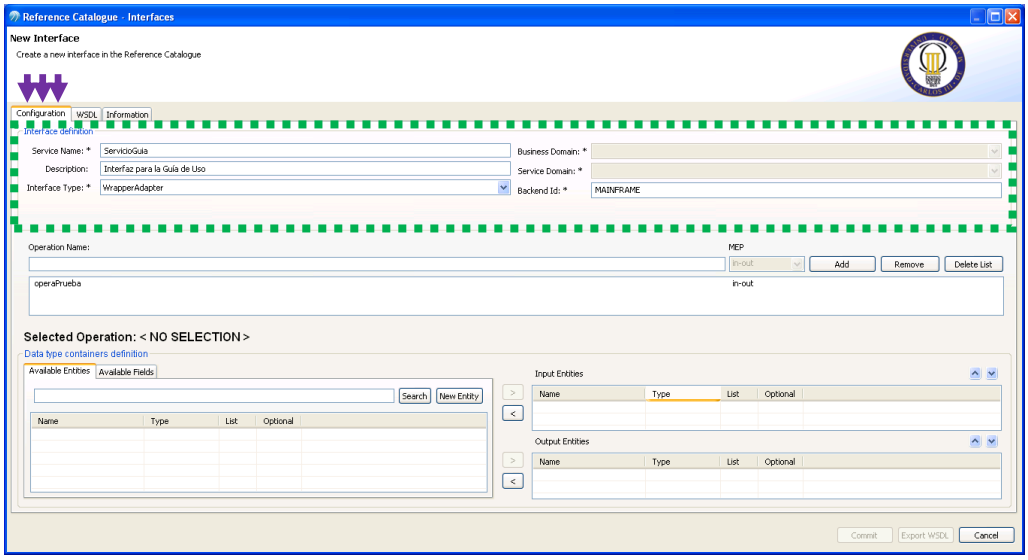
En este apartado se detalla qué información se debe incluir la *Interface definition* para un Wrapper Adapter.

1. Seleccionar del combo ***Interface Type*** el tipo **Wrapper Adapter**.

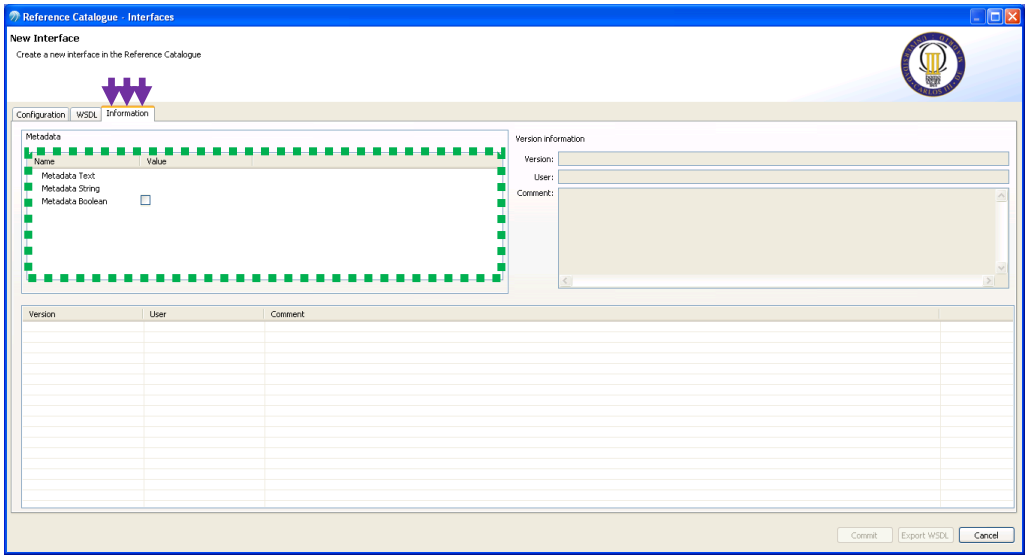


2. Completar los campos que componen la definición de la Interfaz del Wrapper Adapter:

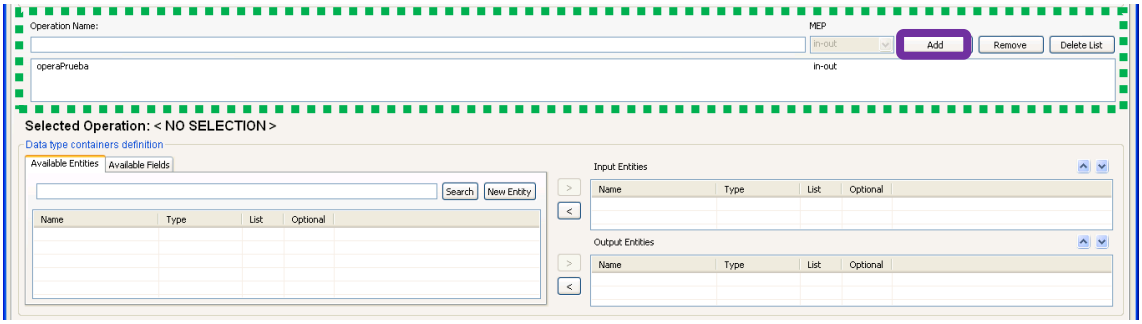
    a) ***Interface Definition**: Service Name* (nombre del servicio), *Description* (definición de la interfaz), *Business Domain* (Business Domain en el que se incluye el servicio), *Service Domain* (Service Domain en que se incluye el servicio) y *Backend Id* (identificador de Backend al que encapsulan).



    b) ***Metadata***: Valores funcionales para los campo definidos en la tabla *Metadata* de la pestaña *Information.*



3. Añadir una operación a la definición de la Interfaz. Para ello, escribir su nombre en ***Operation Name*** y presionar el botón ***Add***. La operación aparecerá en el listado de operaciones incluidas en la Interfaz.



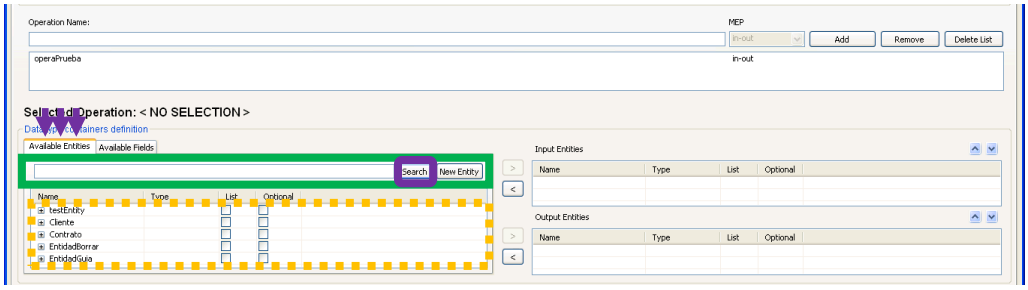
C.3.2.4 Definición Interfaz Entrada/Salida de la Operación

En el siguiente apartado se explica cómo definir la estructura de entrada y salida de la Interfaz a construir. Los pasos que se detallan son comunes para todos los tipos de Interfaces, por lo que aunque el ejemplo se haya constuido con imágenes de la creación de una nueva Interfaz de tipo Channel Adapter, las acciones a realizar son también válidas para el caso de Business Service y Wrapper Adapter.

1. **Definición de la Interfaz de Entrada**. Para definir la Interfaz de Entrada, buscar en la pestaña *Entities,* o en la pestaña Fields de la sección *Data type containers definition* aquellos campos que se deseen incorporar. Seleccionarlos del listado de resultados y presionar el botón “>” para incluirlos en la sección *Input Entities*.

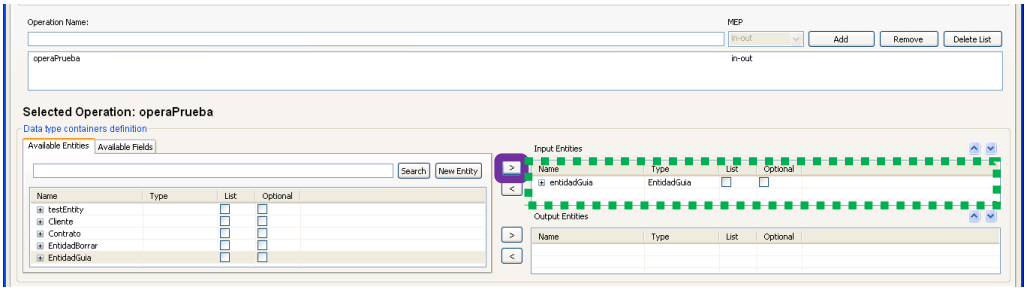
   a) Seleccionar Entity como elemento de la definición de la Interfaz de Entrada.

      a.1) Realizar la búsqueda de la Entidad a incluir desde la pestaña Entities. Para ello, incluir el nombre de la Entidad a filtrar en el campo *Available Entity* y presionar el botón ***Search***.



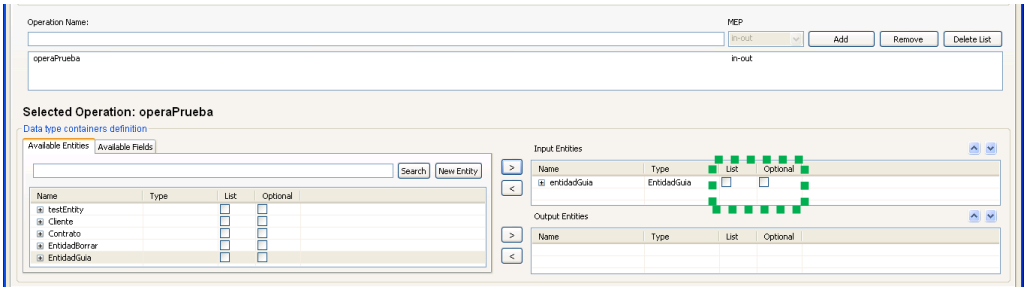
NOTA: Las entidades resultado de la búsqueda pueden expandirse (⊞) para visualizar sus componentes y ser añadidos individualmente.

a.2) Seleccionar, del listado que aparece como resultado de la búsqueda la entidad que se desee y presionar el botón "**>**" para incluirla como elemento de la Entrada de la Interfaz que estamos definiendo.



NOTA: Para quitar una entidad del cuadro de *Input,* seleccionarla y pulsar "**<**".

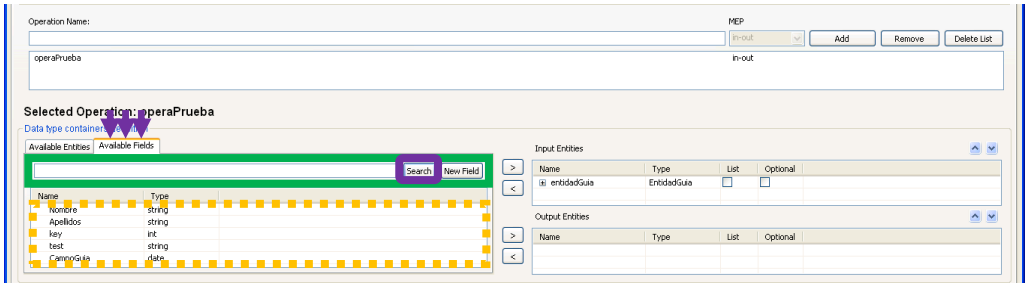
a.3) Una vez la entidad seleccionada aparezca en el listado de campos de la entrada de la Interfaz, podemos definir si la entidad aparecerá como un listado en la Intefaz (check sobre checkbox *List*) o si será opcional (check sobre checkbox *Optional*).



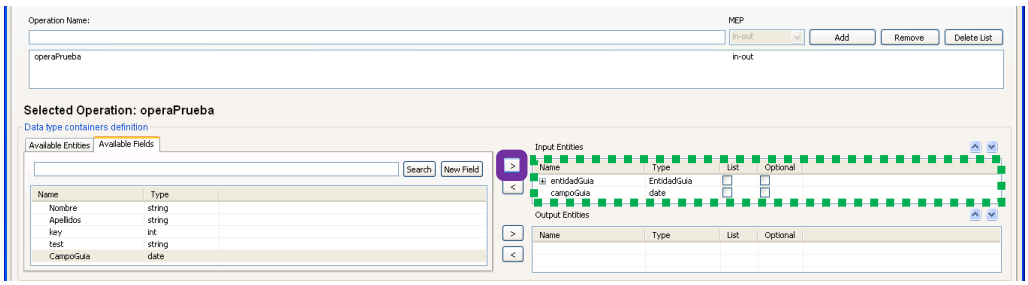
a.4) Repetir el procedimiento para cada una de las entidades que se quieran incluir.

b) Seleccionar Campo como elemento de la definición de la Interfaz de Entrada.

b.1) Realizar la búsqueda del Campo a incluir desde la pestaña Fields. Para ello, incluir el nombre del campo a filtrar en el campo Available Fields y presionar el botón Search.

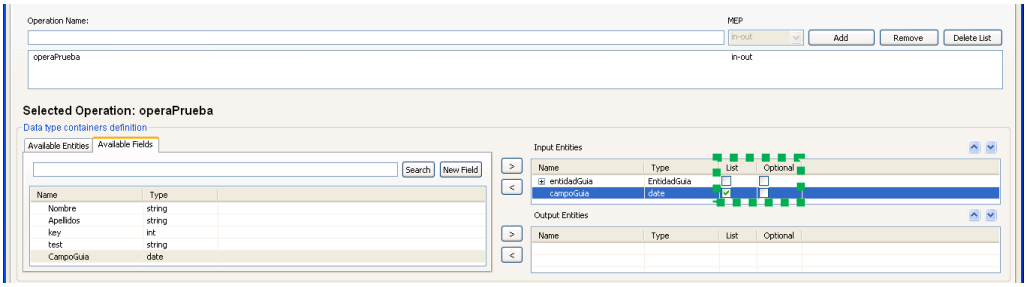


b.2) Seleccionar, del listado que aparece como resultado de la búsqueda la entidad que se desee y presionar el botón "**>**" para incluirla como elemento de la Entrada de la Interfaz que estamos kdefiniendo.



NOTA: Para quitar un campo del cuadro de *Input,* seleccionarlo y pulsar "**<**"

b.3) Una vez el campo seleccionado aparezca en el listado de campos de la entrada de la Interfaz, podemos definir si el campo aparecerá como un listado en la Intefaz (check sobre checkbox List) o si será opcional (check sobre checkbox Optional).

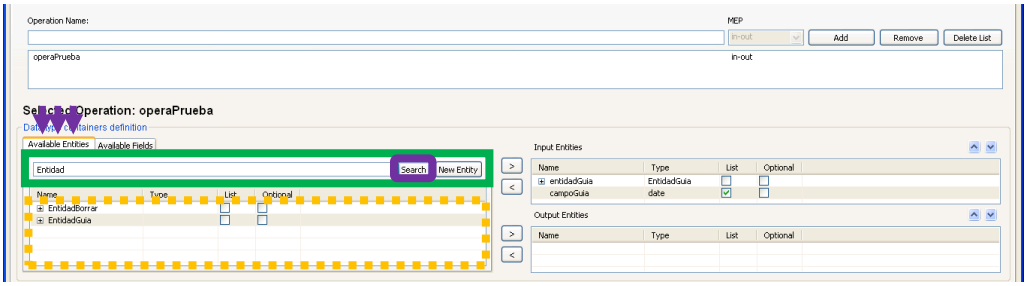


b.4) Repetir el procedimiento por cada uno de los campos que se quieran incluir.

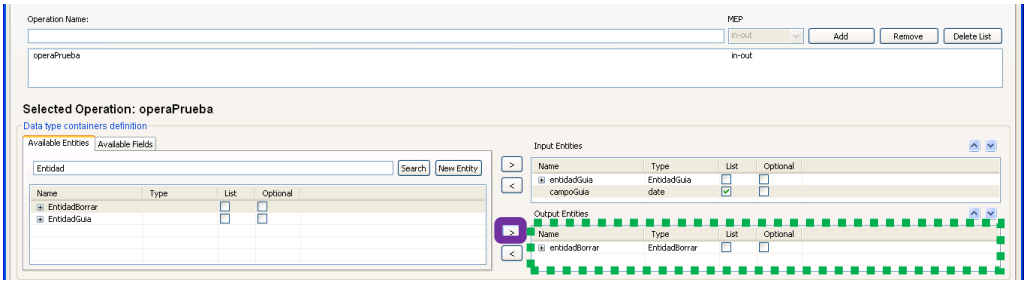
2. **Definición de la Interfaz de Salida**. Para definir la Interfaz de Salida, buscar en la pestaña *Entities,* o en la pestaña Fields de la sección *Data type containers definition* aquellos campos que se deseen incorporar. Seleccionarlos del listado de resultados y presionar el botón ">" para incluirlos en la sección *Output Entities*.

   a) Seleccionar Entity como elemento de la definición de la Interfaz de Salida.

   a.1) Realizar la búsqueda de la Entidad a incluir desde la pestaña Entities. Para ello, incluir el nombre de la Entidad a filtrar en el campo *Available Entities* y presionar el botón ***Search***.

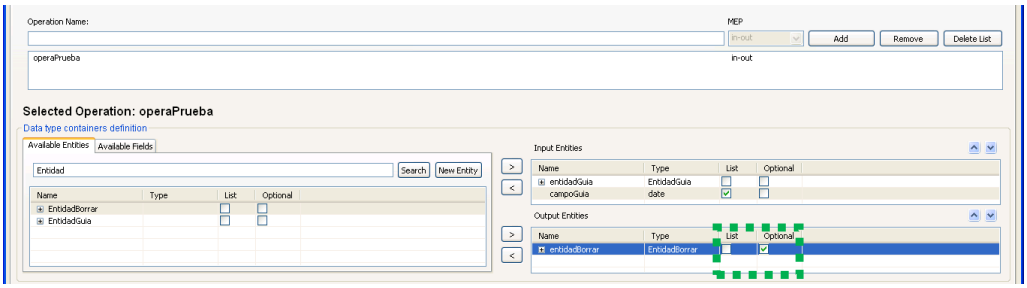


a.2) Seleccionar, del listado que aparece como resultado de la búsqueda la entidad que se desee y presionar el botón "**>**" para incluirla como elemento de la Salida de la Interfaz que estamos definiendo.



NOTA: Para quitar una entidad del cuadro de *Output Entities,* seleccionarla y pulsar "**<**".

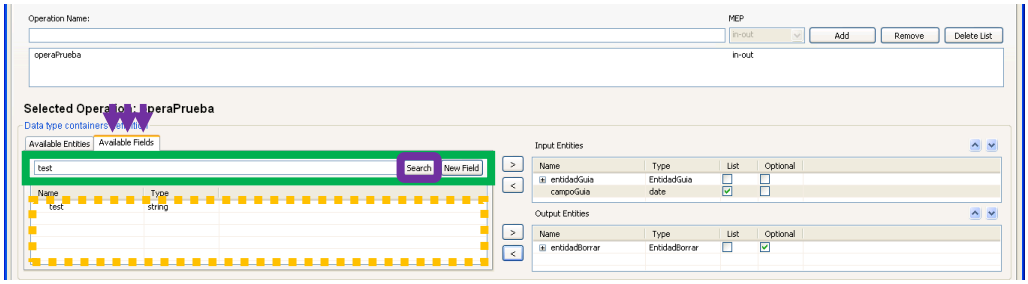
a.3) Una vez la entidad seleccionada aparezca en el listado de campos de la salida de la Interfaz, podemos definir si la entidad aparecerá como un listado en la Intefaz (check sobre checkbox *List*) o si será opcional (check sobre checkbox *Optional*).



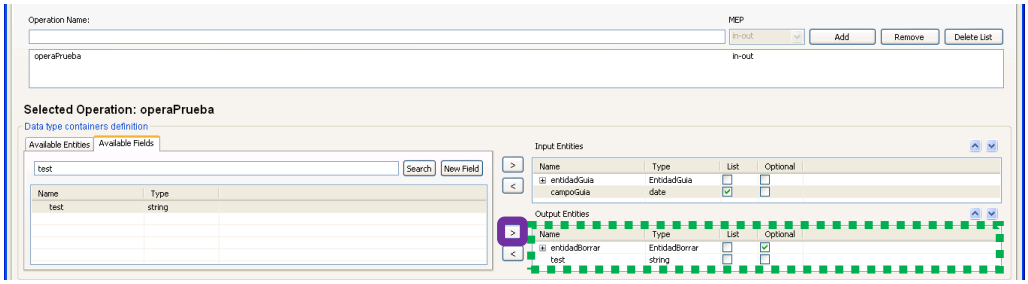
a.4) Repetir el procedimiento por cada una de las entidades que se quieran incluir.

b) Seleccionar Campo como elemento de la definición de la Interfaz de Salida.

b.1) Realizar la búsqueda del Campo a incluir desde la pestaña Fields. Para ello, incluir el nombre del campo a filtrar en el campo *Available Fields* y presionar el botón ***Search***.

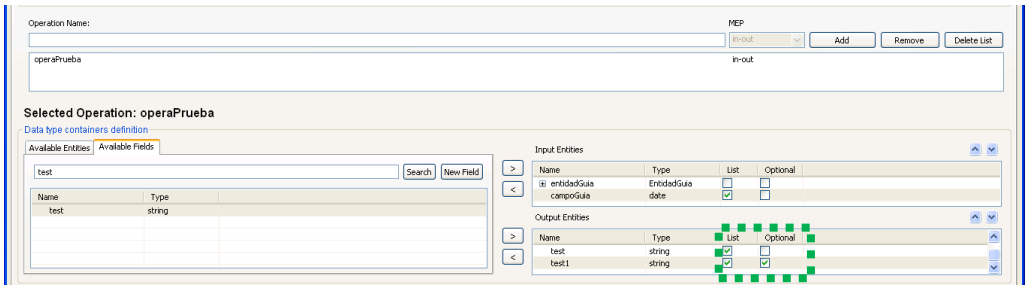


b.2) Seleccionar, del listado que aparece como resultado de la búsqueda la entidad que se desee y presionar el botón "**>**" para incluirla como elemento de la Salida de la Interfaz que estamos definiendo.



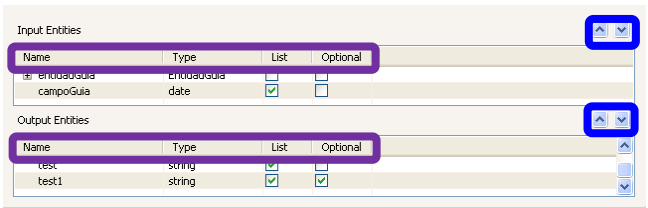
NOTA: Para quitar un campo del cuadro de *Output Entities,* seleccionarlo y pulsar "**<**".

b.3) Una vez el campo seleccionado aparezca en el listado de campos de la salida de la Interfaz, podemos definir si el campo aparecerá como un listado en la Intefaz (check sobre checkbox List) o si será opcional (check sobre checkbox Optional).



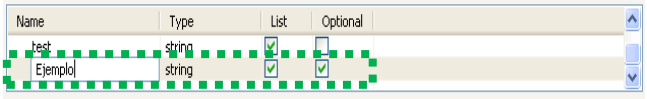
b.4) Repetir el procedimiento por cada uno de los campos que se quieran incluir.

3. **Ordenar los campos** según el orden deseado: se pueden realizar ordenaciones **personalizadas** o **alfabéticas** a través de las cabeceras de los cuadros de Input/Output.

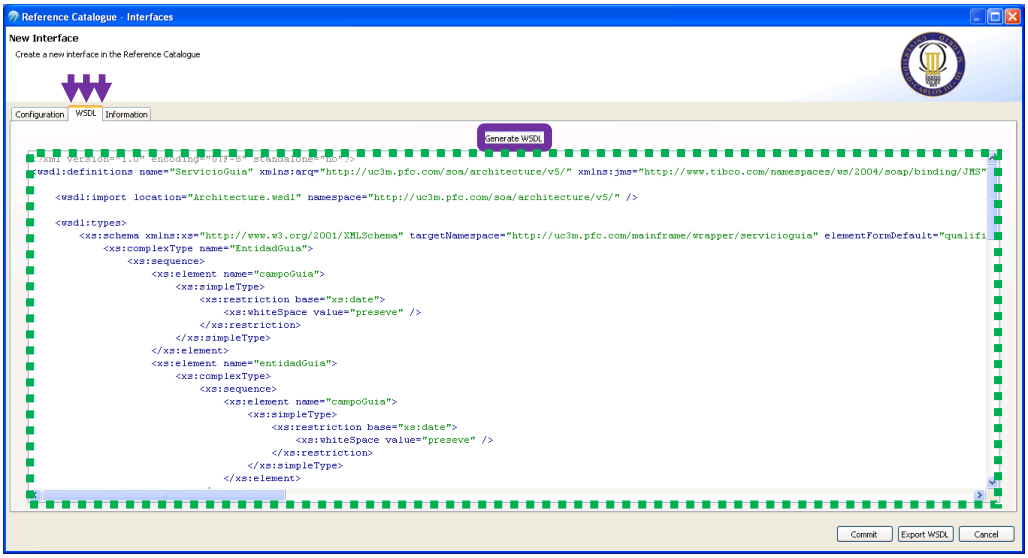


4. Si se desea se pueden **renombrar** los elementos que componen la entidad, de modo que su nomenclatura sea más intuitiva, y permitiendo además la inclusión de

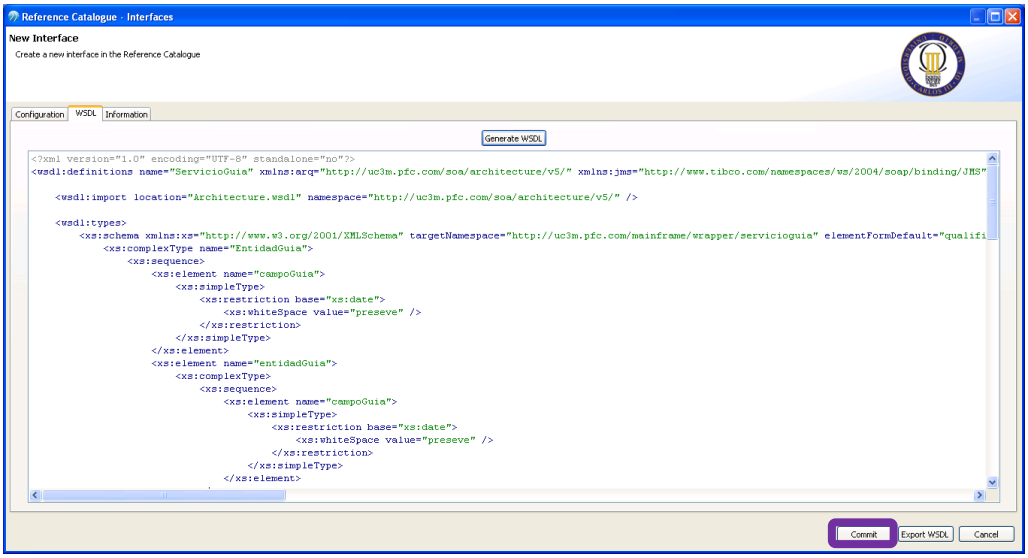
múltiples repeticiones del mismo elemento siempre y cuando el nombre de cada repetición sea distinto.



5. **Visualizar WSDL Interfaz**. Una vez añadidos y modificados los campos/entidades de la interfaz, en la pestaña WSDL del editor, presionar el botón ***Generate WSDL*** para comprobar que la interfaz que se generará es la deseada.



6. Finalmente, presionar el botón ***Commit*** para crear la Interfaz en el Catálogo de Referencia.

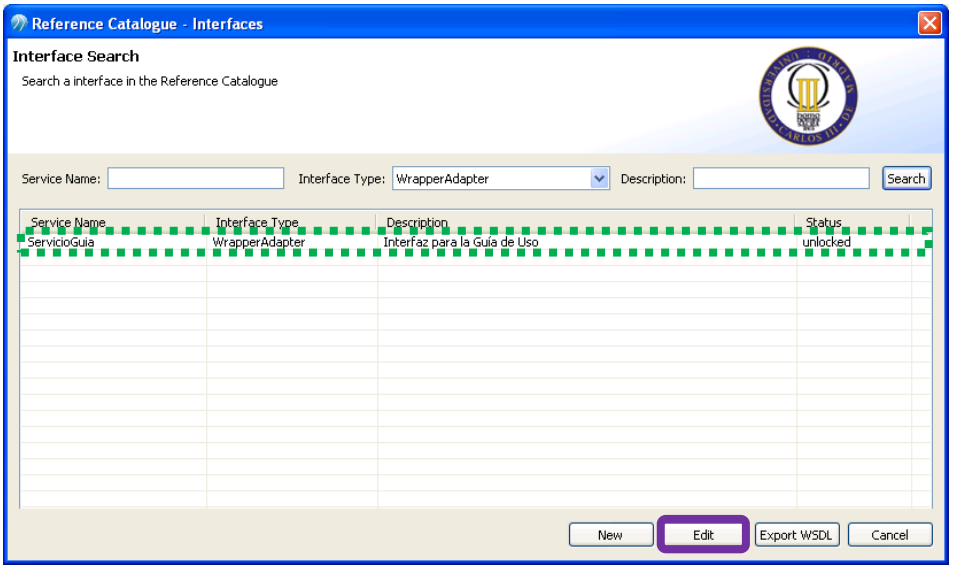


C.3.3 Editar Interfaz en el Catálogo de Referencia

Esta opción de la herramienta *Interface Generator* permite editar una entidad en el Catálogo de Referencia.

1. Realizar la búsqueda de la interfaz que se desea modificar mediante la opción búsqueda de la herramienta *Interface Generator* detallada en el apartado Buscar Interfaz en el Catálogo de Referencia.

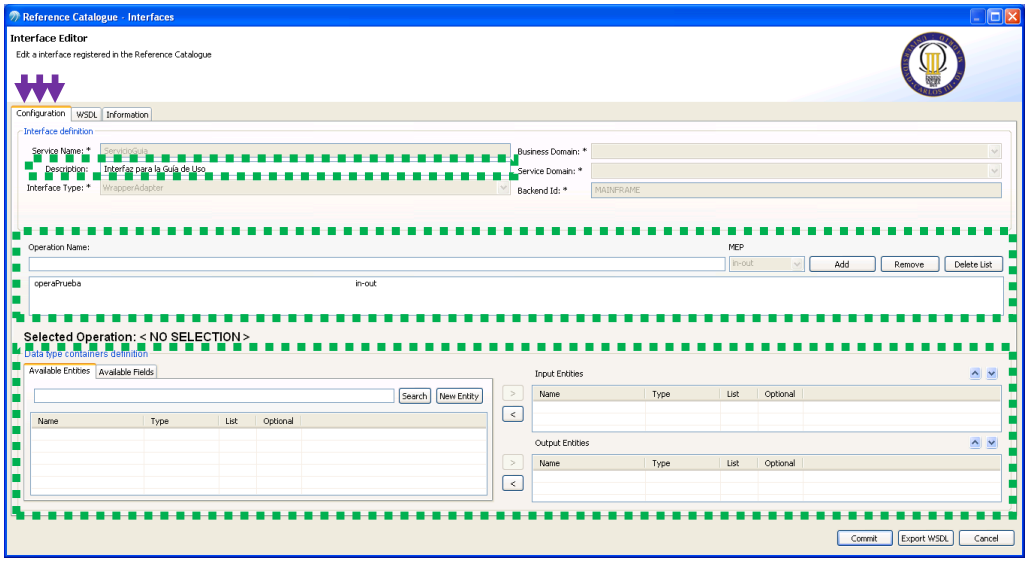
2. Seleccionar del listado la interface a editar y presionar el botón ***Edit*** (también doble click o intro).



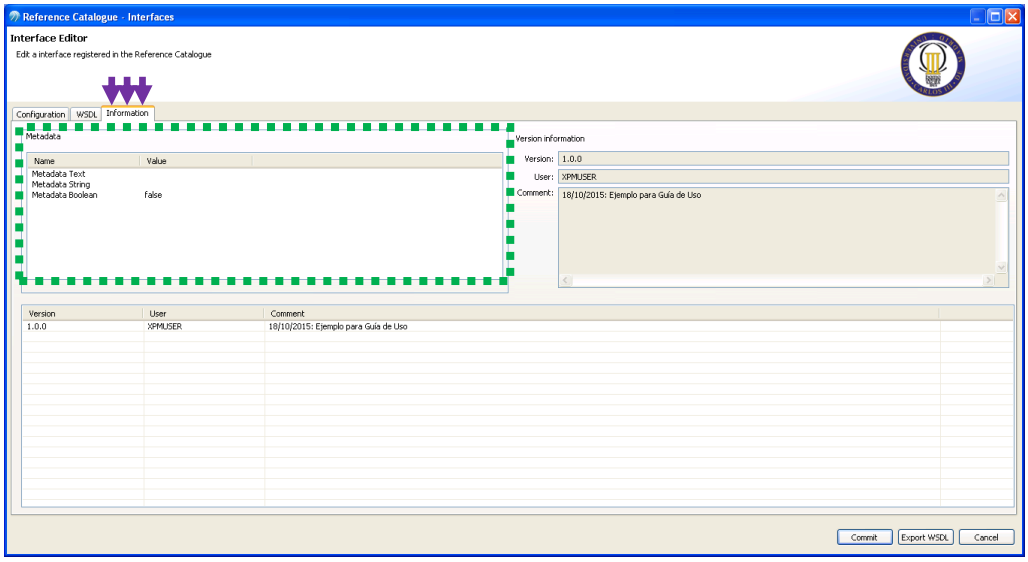
3. Editar la entidad teniendo en cuenta que los campos modificables son:

   a) En el área *Interface definition*: el texto *Description.*

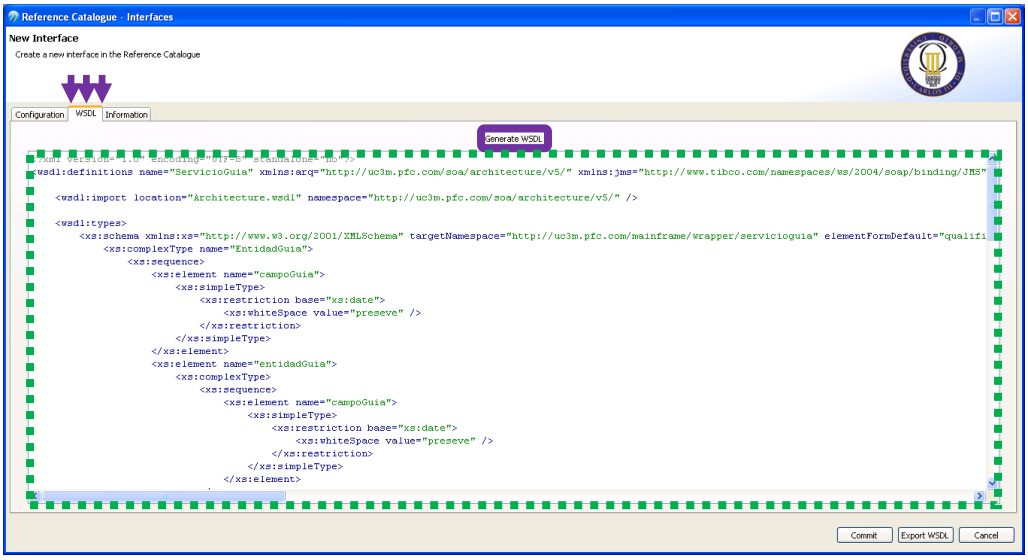
   b) Operaciones y elementos de los mensajes de Input/Output de las mismas (añadir/quitar, nomenclatura, ocurrencias, orden…) tal y como se detalla en Crear Interfaz en el Catálogo de Referencia.



c) Valores de la tabla ***Metadata*** en la pestaña ***Information***.

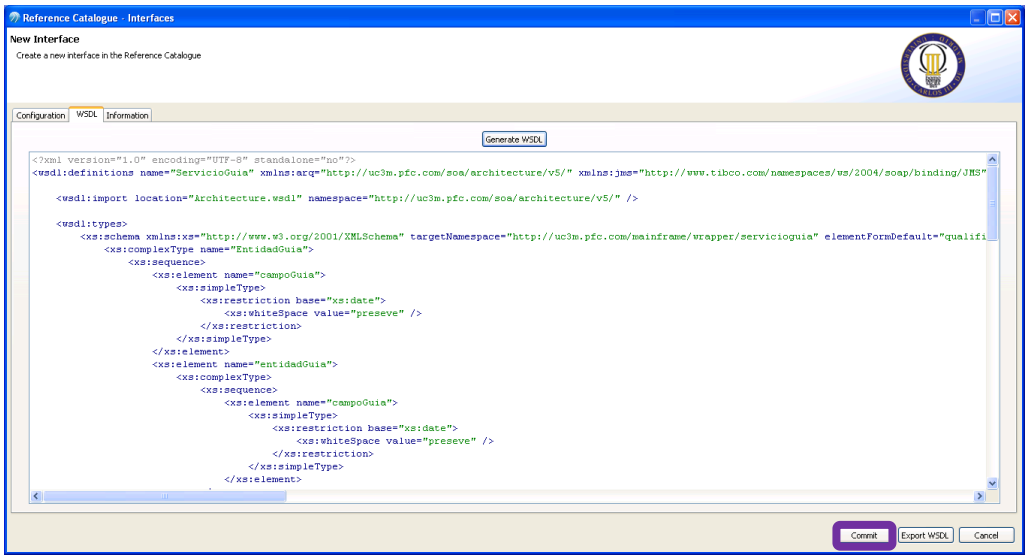


4. Presionar el botón ***Generate WSDL*** para comprobar que las modificaciones en la interface son las deseadas.

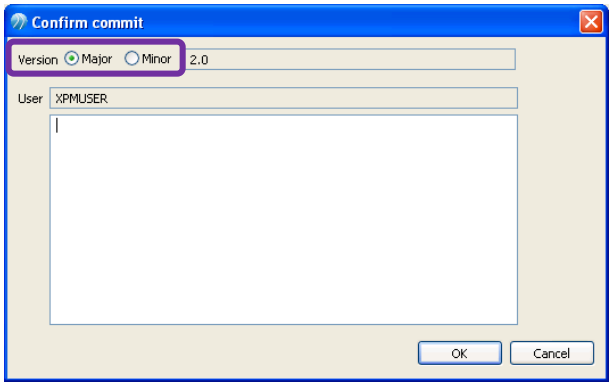


5. Finalmente, aplicar los cambios de la interface sobre Catálogo de Referencia.

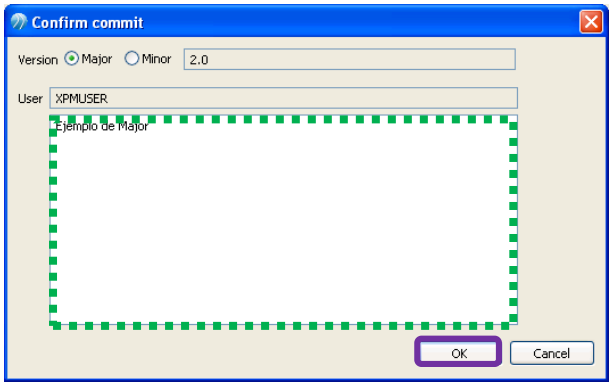
   a) Presionar el botón ***Commit***.



   b) Seleccionar si será una versión ***Minor*** o ***Major***.



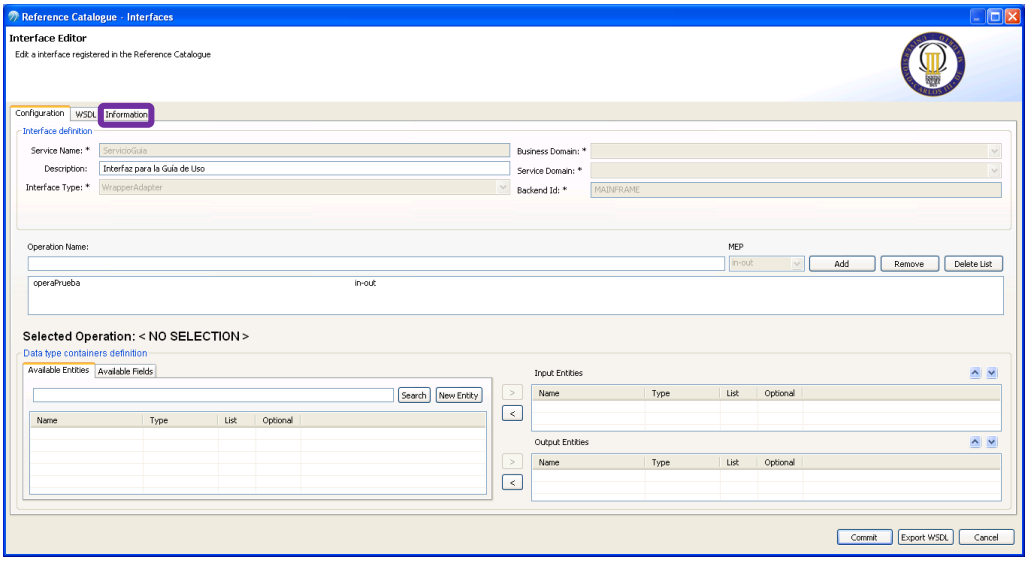
   c) Añadir un ***comentario*** de versión si se desea y ***aceptar*** el registro.



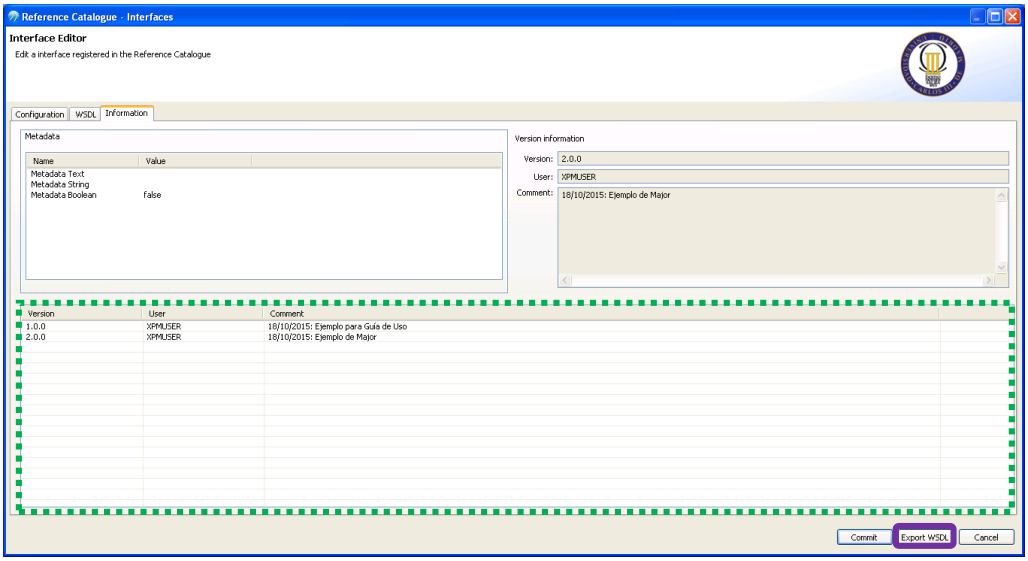
C.3.4 Recuperar Interfaz del Catálogo de Referencia

Esta opción de la herramienta *Interface Generator* permite recuperar versiones anteriores de una Interfaz guardadas en el Catálogo de Referencia. Para recuperar versiones anteriores, sólo puede hacerse editando una Interfaz ya creada.

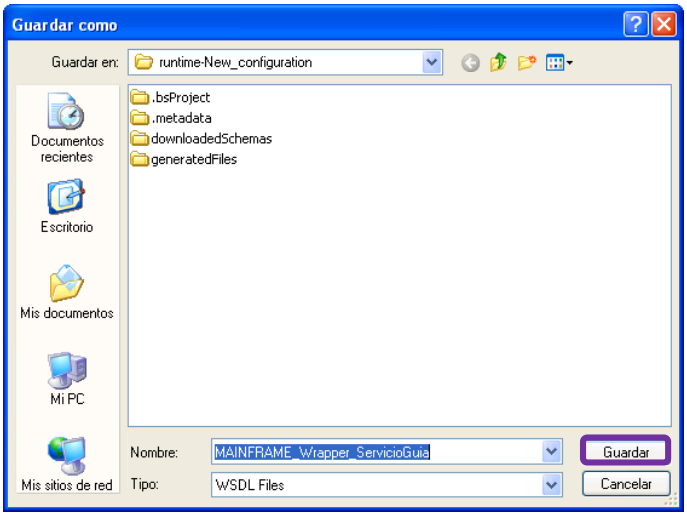
1. Una vez seleccionada una Interfaz para editar y presionar *Edit*, ir a la pestaña ***Information***.



2. Seleccionar la versión que se quiere recuperar de la ***tabla*** y presionar ***Export WSDL*** (también doble click o intro).



3. Finalmente, navegar por el sistema de ficheros para seleccionar la ruta donde se desea descargar la versión anterior de la Interfaz y presionar ***Save (Guardar)***.



C.3.5 Exportar WSDL de un Interfaz desde el Catálogo de Referencia

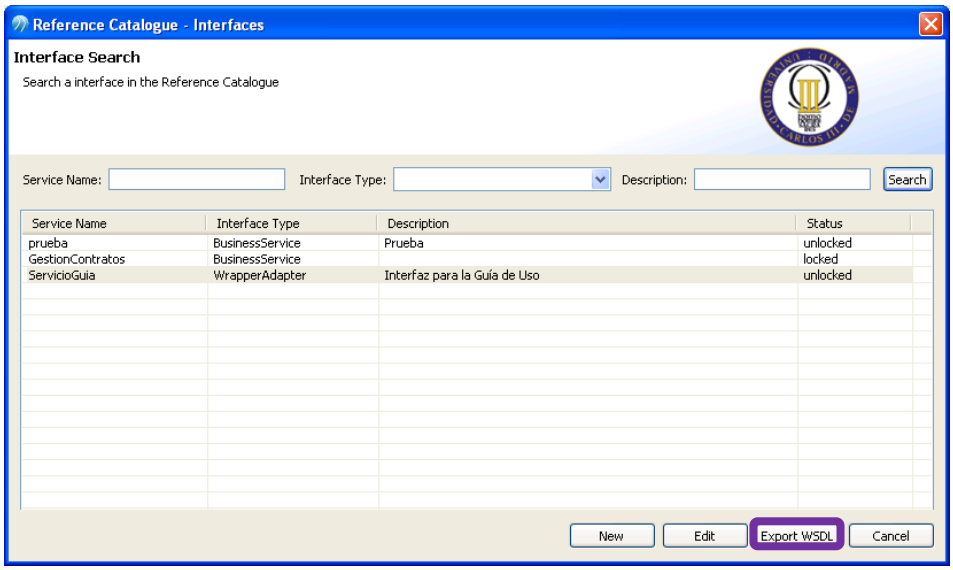
Esta funcionalidad, incluida en la herramienta *Interface Generator*, permite exportar la definición de una Interfaz contenida en el Catálogo de Referencia a un fichero físico ".wsdl", que se almacena en el entorno local.

1. Se puede invocar a la función de exportación de interfaces desde:

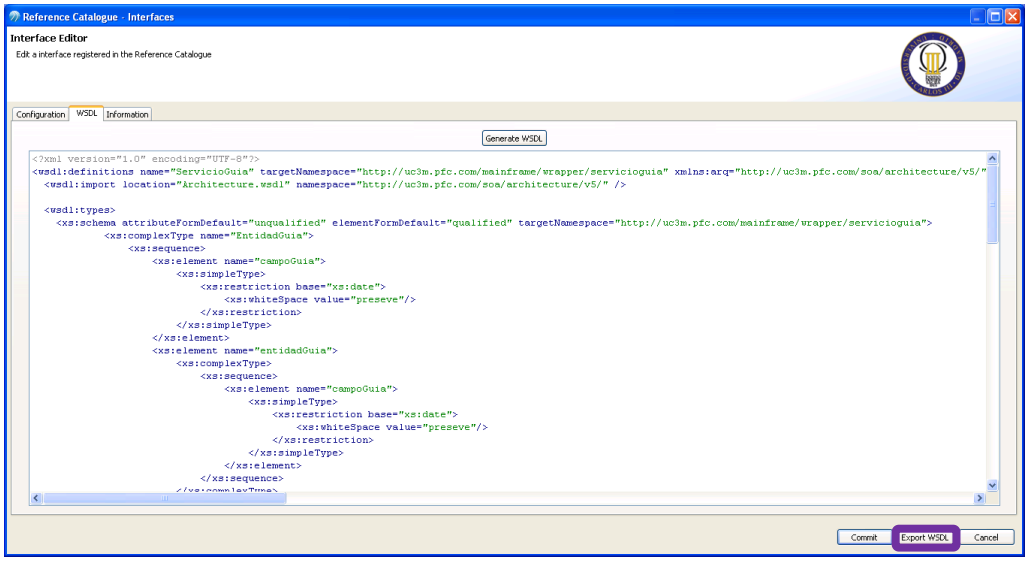
    a) Ventana principal de la herramienta *Interface Generator:*

    a.1) Realizar la búsqueda de la interface que se desea exportar mediante la opción búsqueda de la herramienta *Interface Generator* detallada en el apartado Buscar Interfaz en el Catálogo de Referencia.

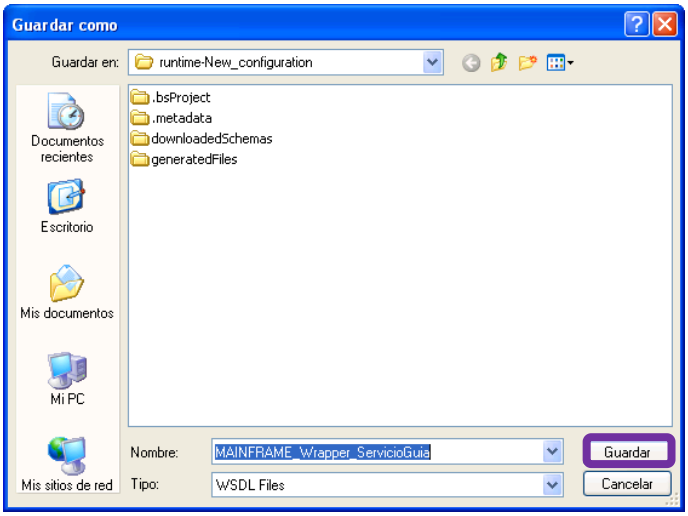
    a.2) Seleccionar del listado de resultados la entidad a exportar y presionar el botón ***Export WSDL***.



b) Ventana de edición de interfaces, *Interface Editor:*



2. Finalmente, aparecerá una ventana de salvado (Save As): una vez seleccionada la ruta y nombre de salvado en local (por defecto, la ruta destino es la del workspace de trabajo y el nombre por defecto cumple con el estándar), presionar ***Save (Guardar)*** para completar la exportación.



Anexo D.

Proyectos Ágiles: Scrum

Las metodologías tradicionales ya mencionaban las prácticas y conceptos utilizados en Scrum, llegando a utilizarlos en mayor o menor medida con el objetivo de mejorar el resultado de los proyectos.

Scrum sitúa todas estas prácticas, identificadas en el estudio de la manera de trabajar de equipos de alta productividad, en el centro del proceso, apoyándose unas a otras y estableciéndose como una filosofía de trabajo.

D.1 Definición de Scrum

La metodología Scrum propone un marco de conceptual de trabajo en el que se aplican de manera regular un conjunto de **buenas prácticas** para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto.

El uso regular de estas prácticas (entregas frecuentes priorizadas por valor, potenciación del equipo de trabajo, mejora continua de producto y de proceso, etc.) permite que los cambios dentro de un proyecto sean aceptados de manera natural y proporciona al cliente margen para flexibilidad e innovación así como productividad y calidad.

D.2 Fundamentos de Scrum

Scrum define un marco conceptual de trabajo basado en las siguientes prácticas:

- El desarrollo incremental de los requisitos del proyecto en bloques temporales cortos y fijos (iteraciones de un mes natural y hasta de dos semanas, si así se necesita).
- La priorización de los requisitos por valor para el cliente y coste de desarrollo en cada iteración.

- El control empírico del proyecto. Por un lado, al final de cada iteración se demuestra al cliente el resultado real obtenido, de manera que pueda tomar las decisiones necesarias en función de lo que observa y del contexto del proyecto en ese momento. Por otro lado, el equipo se sincroniza diariamente y realiza las adaptaciones necesarias.
- La potenciación del equipo, que se compromete a entregar unos requisitos y para ello se le otorga la autoridad necesaria para organizar su trabajo.
- La sistematización de la colaboración y la comunicación tanto entre el equipo y como con el cliente.
- El *timeboxing* de las actividades del proyecto, para ayudar a la toma de decisiones y conseguir resultados.

D.3 Aplicación de Scrum: El Proceso

En *Scrum* un proyecto se ejecuta en bloques temporales cortos y fijos (iteraciones de un mes natural y hasta de dos semanas, si así se necesita). Cada iteración, también denominada *Sprint*, producirá un entregable que represente un incremento funcional sobre la solución final.

El proceso parte de la lista de requisitos priorizada del producto, que actúa como plan del proyecto. En esta lista el cliente prioriza los objetivos balanceando el valor que le aportan respecto a su coste y quedan repartidos en iteraciones y entregas. De manera regular el cliente puede maximizar la utilidad de lo que se desarrolla y el retorno de inversión mediante la replanificación de objetivos del producto, que realiza durante la iteración con vista a las siguientes iteraciones.

D.3.1 Planificación de la iteración

Al comienzo de la iteración se realiza la reunión de planificación, en la que se perseguirán las siguientes tareas:

1. **Selección de requisitos** (4 horas máximo). El cliente presenta al equipo la lista de requisitos priorizada del proyecto. El equipo pregunta al cliente las dudas que surgen y selecciona los requisitos prioritarios que se compromete a completar.
2. **Planificación de la iteración** (4 horas máximo). El equipo elabora la lista de tareas necesarias para desarrollar para alcanzar su compromiso. La estimación de esfuerzo se hace de manera conjunta y los miembros del equipo se autoasignan las tareas.

D.3.2 Ejecución de la iteración

Diariamente el equipo realiza una **reunión de sincronización** (15 minutos máximo). Cada miembro del equipo inspecciona el trabajo que el resto está realizando (dependencias entre tareas, progreso hacia el objetivo de la iteración, obstáculos que pueden impedir este objetivo) para poder hacer las adaptaciones necesarias que permitan cumplir con el compromiso adquirido.

D.3.3 Inspección y adaptación

Al final de la iteración se realiza una reunión de revisión de la misma:

1. **Demostración** (4 horas máximo). El equipo presenta al cliente los requisitos completados en la iteración, en forma de incremento de producto preparado para ser entregado con el mínimo esfuerzo. En función de los resultados mostrados y de los cambios que haya habido en el contexto del proyecto, el cliente realiza las adaptaciones necesarias de manera objetiva, ya desde la primera iteración, replanificando el proyecto si fuese necesario.
2. **Retrospectiva** (4 horas máximo). El equipo analiza cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente, mejorando de manera continua su productividad.